

SIMULATION OF QUEUEING STRUCTURES

—
a feasibility study

by
Berth Eklundh
—



Department of Telecommunication Systems
Lund Institute of Technology

LUND
SWEDEN

1982

Simulation of Queueing Structures

a feasibility study

AKADEMISK AVHANDLING

som för avläggande av teknisk doktorsexamen
vid tekniska fakulteten vid Universitetet i
Lund kommer att offentligen försvaras å
E-sektionen, hörsal E:B, fredagen den 19
november 1982 kl 10.15

av

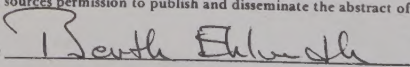
Berth Eklundh
civ ing, Sml

Organization LUND UNIVERSITY		Document name DOCTORAL DISSERTATION	
Department of Telecommunication Systems P O Box 725, S 220 07 LUND, Sweden		Date of issue October 26, 1982	
Author(s) Berth Eklundh		CODEN: LUTEDX/(TETS-1002)/1-180/(1982)	
		Sponsoring organization Telefonaktiebolaget LM Ericsson	
Title and subtitle Simulation of Queueing Structures - a feasibility study			
Abstract. Discrete Event Simulation can be used to model technical systems with virtually unlimited accuracy. Restrictions are imposed, however, mainly by man himself and his ability to handle large and complex models and programs, and by computer facilities. The restrictions imply that some system properties have to be left out of the model, and that the problem of selecting properties to incorporate appears. This selection is governed by the performance to be investigated by simulation, and the dissertation begins with a discussion of the capabilities of simulation modelling and of the results possible to obtain by simulation. It is found, that the vast applicability of Discrete Event Simulation makes it hard to state anything definite about the process of selection. The perspective is therefore narrowed somewhat and the concept of queueing structures is introduced and explained. Results obtainable by means of these concepts are then related to simulation and analytical methods. The subjectivity of the goodness of the results is stressed and the methods available for increasing model confidence are discussed. In order to demonstrate the evolution of an ordinary simulation project, an extensive example is related. The example, the control system of a computerized telephone exchange, is described in some detail with respect to the modelling issues, and some numerical results are shown. The structure of simulation modelling is then analysed in order to find the fundamental properties of the modelling process. It is found that three concepts can be seen to describe modelling and to guide the unexperienced modeller. These concepts are used to form a modelling scheme and the scheme is applied to the example mentioned above. The models resulting from the application of the scheme are in many respects simpler than the model that emerged from the unrestricted modelling and numerical results are presented to clarify the small impact of the simplifications. The modelling scheme is thereafter concentrated and presented as a step by step guide.			
Key words Modelling, Simulation, Discrete Event, Modelling Scheme, Queueing Structures, Performance Analysis			
Classification system and/or index terms (if any)			
Supplementary bibliographical information		Language English	
ISSN and key title		ISBN	
Recipient's notes		Number of pages 180	Price
		Security classification	

Distribution by (name and address) Department of Telecommunication Systems,
P O Box 725, S 220 07 LUND, Sweden. Telephone 046-109011

I, the undersigned, being the copyright owner of the abstract of the above-mentioned dissertation, hereby grant to all reference sources permission to publish and disseminate the abstract of the above-mentioned dissertation.

Signature



Date October 26, 1982

SIMULATION OF QUEUEING STRUCTURES

a feasibility study

by
Berth Eklundh

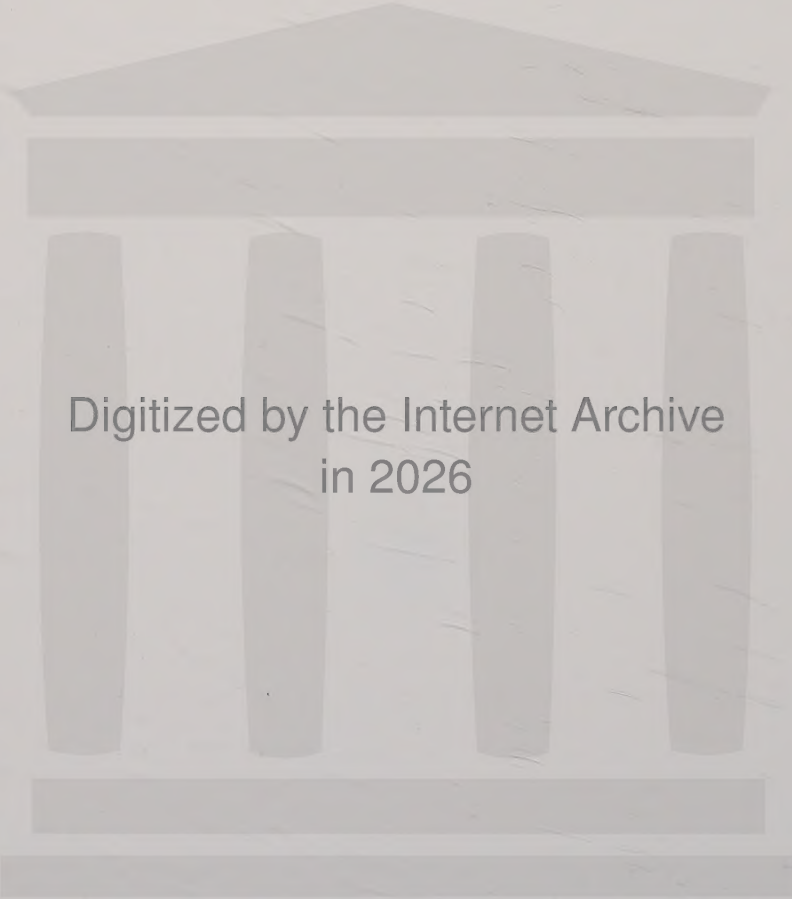


Department of Telecommunication Systems
Lund Institute of Technology
LUND
SWEDEN

CODEN: LUTEDX/(TETS-1002)/1-139/(1982)



和尚打鐵



Digitized by the Internet Archive
in 2026

Preface

The contemporary society is probably the most complex that has ever existed. The invention of various technical equipment has since the beginning of the industrial revolution made it harder and harder for man to understand the world he is living in. The social and economical structures that have developed during the last century, are far too complex for man to understand by just experiencing them. These structures are not man-made in the sense that they are built from scratch with a certain goal. They have developed as the joint effect of many different ventures, and there is no position within the systems from which anyone can get an overall view. They have, in this respect, so far differed from specific technical devices invented and developed by man. Man-made things have also been possible for man to understand, and even if not everybody has been able to understand how technical inventions work, there has always been someone who has had enough knowledge of the machine or system to be able to tell its function. This is beginning to change, however. With the advent of computers and computerized functions, it has become possible for man to construct and build systems that behave in an unexpected way which no one can really explain. This may seem to be quite mysterious, but it is the effect of many co-operating functions, each of which is quite possible to understand, but whose joint effect is hard to predict. In this respect they are very similar to the social and economic structures mentioned above. Except for the unpredictable behaviour of each individual that contributes to the changes of our economy, we know quite well how an economic system behaves when we look at one function at a time, but when these are put together and are allowed to interact, we are at a loss and find that we can say very little about the overall performance of such an economic system.

Man has, probably since he first began to think about the world he lives in, tried to form some kind of explanation of the things that are essentially impossible to understand. He has developed various philosophical systems, which almost all have had the deficiency that they had to start somewhere, at a point that later often proved to be not quite as solid and indisputable as the philosopher had believed. At first science was closely related to religion, and the battles that was fought between for example the church and some of the scientists during the inquisition are well known. Later science tried to rid itself of the dogmas of the church and to explain cosmos in a purely scientific way. At certain times man has been convinced that he was close to the final knowledge of a restricted area, as for example in the late nineteenth century when he believed that all major problems in physics were solved.

All these attempts to explain something essentially inexplicable, whether God or Quantum Mechanics, have something in common: they can all be looked upon as metaphores, pictures or models. They try, usually by simplification, to explain something that is too complex for man to understand.

The word model has very wide connotations; it is used together with all kinds of processes where some simplification takes place, and denotes

sometimes a very conscious process while at other times it denotes simplifications that we do in everyday life. Both knowledge and models can be viewed as a sort of generalisation, even if modelling has come to be more related to an active process in which the generalisations involved are scrutinised more closely. To be able to say anything at all about a specific, not trivial, phenomenon, we have to generalise. The question is only when a generalisation is permissible and how far it can be extended. The answer is probably that it depends. It depends on the kind of and on how much information we have at hand when we generalise. If a person, after one single trip on a train, where he had to share compartment with a lady with a dog, said that you always meet dogs on trains, we would not be inclined to agree. But if someone said to us that he always gets jammed in traffic on his way home from work on Friday afternoon, we would not dispute such a statement. So whether we allow people to generalise or not, depends on the kind of information they possess and on how they have gained it. But the way, in which man gains, retrieves, and stores information, is not understood. All we know is that one way or another, we are able to investigate our environment and extract pertinent information about it. Cognitive psychology gives us a few hints, but can not tell us how we sort information and why we choose to remember some information while we forget other. What we can find however, is that information is processed according to some scheme, and that this scheme acts as a filter that selects information that we believe to be of importance. This observation will be of great importance in what follows.

As stated above, we make generalizations based on the information we get from outside, and if the generalisation process is conscious and follows a specific scheme we often talk of the result as a model. A scientific model is of little use however, if it cannot give results that can be compared with "reality" to assure us that the model is an accurate description of that "reality". Things that could be compared are usually expressed as numbers.

A good model has to be possible to manipulate. Mathematical expressions can, due to the concentrated formalism and the vast accumulated mathematical knowledge, be manipulated to yield amazingly clear and simple results. But not all phenomena lend themselves to be expressed as mathematical relations.

No one would ever dream of trying to express the relation of Rosalind and Orlando by a mathematical formula, but also more simple and prosaic relations can be hard to grasp as mathematical expressions.

But mathematics is not the only tool available for analysis of systems. Another method, that has become useful and tractable, due to the rapid development of computers in the last decades, is the method of simulation. Simulation means to perform experiments with a model that has been fed into a computer via some programming language. There exists a number of different simulation techniques and applications, but the more formal definitions will come later in the text. With simulation we are able to treat our models virtually as we like, but development of the model and manipulation of it contain many traps and sources of error, and it is the aim of this study to investigate the merits and disadvantages of the simulation technique with this taken into account.

Acknowledgements

A list of acknowledgements is like eastern diplomacy: it is not what is present but what is missing that counts. Everyone should be convinced however, that any regretful mistake made here what that matter is concerned, is accidental only, and that no one has been left out on purpose.

First of all it is my great pleasure to thank my adviser, Prof. Bengt Wallström, for his help and encouragement during the conduct of this research. Without his willingness to let me go as I pleased, and his ability to tell me when I talked too much about petty details, nothing of this would have been possible.

The personnel of the Department of Telecommunication Systems deserve thanks for the stimulating atmosphere provided by work and social contacts in good proportions. Special thanks go to Bengt Stavenow and Ulf Körner for their labor with my long – and at times surely tedious – manuscript. I hope you will recover in due time. I am also indebted to Jan Karlsson for valuable discussions on subtle simulation issues.

As for absent friends, I am happy to express my appreciation to Dr. Arne Nilsson, formerly at the department, currently at North Carolina State University, for having introduced me to the subject and encouraged me to pull myself together and come up with the present text.

All the curves of the dissertation are made by Lena Svensson, and although impeccable I have at times thought of having one of her beautiful drawings of a twisted pine included instead of the curves, but that would perhaps have been less appropriate. I am at any rate most grateful.

Monica Bäckström typed the appendix, and anyone who reads it will know what that must have been like. Thank you.

Not even researchers can live on nice atmosphere and enthusiasm. The Ericsson company has supported me greatly by providing the Department with the funds necessary for my research. This is indeed generous in its own right, but people within the company have, moreover, made it their personal business to help me succeed. I would like to thank just a few:

Lars-Olof Norén, for your vast technical insights and your ability to grasp new ideas and trust young people.

Klaus Wildling and Tomas Karlstedt for your never ending knowledge about AXE and for your help during the time when I badly needed seemingly unobtainable information.

The personnel of XF/YG, and especially Björn Fulton and Esbjörn Malcus, for help with printing, layout and figures.

Our entire life is a process of learning, and we learn more from some than from others. My skeptic view of the world, and of our ability to make any definite statements about it, may appear here and there in the text. For this view, which I have gained during endless discussions with my

true friends and for which I am indeed grateful, I have to thank you Kent and Rikard.

Finally, I am of course indebted to you: Carin and Karl Erik.

Abstract

Technical and other systems are becoming increasingly complex. Our ability to understand how they work and perform is as a consequence decreasing and we therefore try to simplify their complex appearance.

Models are widely used for description of various phenomena which are hard to investigate without simplification. A model is a description of a system which, if it is good, comprises the important properties of a system and which can be used to deduce information about the original system.

Such models are usually implemented and evaluated, and there are two main techniques for this purpose: analytic methods, as mathematics and other formal methods, and simulation.

The thesis deals with the simulation application, and reviews the feasibility of simulation experiments as a means for analysing complex systems with respect to their performance.

Simulation can be applied to a large number of different models and the thesis is, in all but the first chapters, restricted to applications which can be described as queueing structures.

This restriction implies that the performance of a system can be expressed in terms of waiting times, queue lengths and similar measures. Once this restriction is made, the feasibility of simulation studies of queueing structures with respect to common performance measures is investigated. After a general discussion about what models are and how they are used, some of the presuppositions of the thesis are stated. These comprise definition of queueing structures, an investigation of results obtainable with other methods than simulation and a survey of results possible to obtain by simulation studies.

The credibility of the obtained results depends on several factors. Verification, validation and other arguments for models being credible are investigated and related to the special circumstances prevailing in the thesis. It is stated that simulation models and results are seductive and that results usually are less credible than expected.

The limited credibility implies that simpler models could be used, but it is observed that there is a lack of guidelines for the modelling process. Guidelines which could aid the modeller and make him incorporate important system behaviour in the model and exclude unimportant. The main purpose of the thesis is to investigate whether such guidelines could be found.

In order to demonstrate what is meant by important properties, an extensive example is related. This example is not the sole example investigated to find out which these properties could be, but it is used because it serves as a good illustration of the points that follow.

The example is a computerized control system: a part of a modern telephone exchange. The system, how it is modelled, and the structure of the resulting program are described. Some results, which show the main performance of the system, are related and discussed.

The structure of simulation modelling is thereafter investigated, and it is described how three important concepts can be used in the modelling process. These concepts — model balance, deterministic analysis and operational analysis — are explained and applied to the formation of a modelling scheme. The scheme is at first described in general terms and various aspects of modelling are related to the restrictions imposed by the scheme. The previous example is remodelled by use of the modelling scheme and it is shown how simpler models, which are as credible as the original model, can be formed. Two additional models, each with an increased degree of simplification, are presented. Results obtained from these models are compared with results obtained from the original model.

The modelling scheme is finally concentrated and a step by step guide to queueing structure modelling is described.

The thesis concludes with some general remarks and some recommendations for future research.

Contents

Preface

1. Introduction

1.1 Outline of the study

1.2 Object

2. Models

2.1 General

2.2 Analytical models

2.3 Simulation models

2.3.1 The simulation technique

2.3.2 Different approaches

2.4 Accuracy of models

3. Preliminaries and presuppositions

3.1 Queueing structures

3.1.1 General

3.1.2 Queueing structure concepts

3.2 Analysis of queueing structures

3.3 Obtainable results

4. Verification, validation and credibility

4.1 General

4.2 Verification of Simulation Programs

4.3 Validation of Simulation Models

4.4 Credibility of models

5. A Conventional Approach

5.1 A specific task

5.2 The AXE system

5.3 Modelling the AXE system

5.4 A simulation model

5.5 Verification and Validation

5.5.1 Verification

5.5.2 Validation

5.6 Results

6. A simplification scheme

6.1 General

6.2 Model Balance

6.3 Deterministic Analysis

6.4 Operational Analysis

6.5 Balanced models; Deterministic and Operational Analysis

7. An alternative approach

7.1 General

7.2 A simplified AXE model

7.2.1 Deterministic Analysis

7.2.2 Operational Analysis

7.2.3 Model Balance

7.3 Models and Results

8. The non-specialist approach

8.1 The Simulationist's Handbook?

8.2 A Modelling Scheme

9. Conclusion

10. References

11. Appendix

1. Introduction

1.1 Outline of the study

The present text is the result of several years of intense simulation studies and is organized in nine chapters, each discussing some specific problems that arise when simulation is used to investigate complex systems. Simulation does not belong to the exact sciences, and research can therefore neither be conducted nor conveyed by formal methods. The presentation can consequently be a bit "verbose" at times, but there does not seem to be any other way to describe the modelling process. The reader is encouraged to read the thesis as a mosaic or a jigsaw puzzle in which every piece has a place, but in which the pieces do not appear in perfect order. It is likely that the pieces of the thesis could have been rearranged, but it is not obvious that the rearranged pattern would have been any easier to understand and read. With these reservations, the outline is as follows:

The introduction describes the outline of the study and the object. Simulation, what it is and how it can be used, is discussed in general terms.

The second chapter goes a bit beyond the main aim of the study and investigates modelling more generally. Some specific modelling techniques, analytic modelling and simulation modelling, are discussed generally. The subject is thereafter narrowed somewhat, and simulation, especially Discrete Event Simulation, is scrutinised more closely. The potentials of the technique are described and a section about the accuracy of models ends the chapter.

Chapter three begins by stating that the applications of simulation are vast and that in order to be able to make a meaningful investigation, the area of application has to be limited. This is done in this chapter and the concept of queueing structures is introduced. Some means to analyse such structures are described, and special attention is paid to analytic models and simulation models. A section about the results that can be obtained from simulation models concludes the chapter.

Models are of no use unless they are correct in some sense. The fourth chapter deals with what is meant by a "correct" model and how to accomplish it. As for simulation models, three tools can be identified: verification, validation and credibility. Verification deals with program and model agreement, validation with whether the model and the "real" world show the expected resemblance and credibility with whether we can have confidence in results produced by the model and the program. These issues are discussed and related to simulation of queueing structures.

A specific simulation task, simulation of the AXE exchange system, is described in chapter five. The description is incorporated in the thesis because without concrete examples it is extremely hard to discuss model-

ling and convey experience. After a general system description, modelling of the AXE system is described. The presentation focuses on the generality of the modelling and very little program listings are shown. Model and program development are described and the verification and validation processes are related. Finally, some results from the simulation are presented.

Chapter six is devoted to an investigation of the structure of simulation models and is perhaps the most important chapter of the thesis. The investigation of structure results in a proposal of a modelling scheme which can be used in the early stages of a modelling process. The scheme employs three concepts: model balance, deterministic analysis and operational analysis. These concepts are put together to form the scheme and it is described how to put the scheme to use.

The AXE system remodelled is in chapter seven by using the proposed modelling scheme. Two alternative models, with different degrees of simplification, are presented and compared with the model of chapter five in order to evaluate the effects of the simplifications. The individual steps of the scheme are described and the AXE system is expressed in terms relevant to the scheme. Results from the alternative models are presented as curves showing relevant performance measures and comparisons are also made with analytic results.

Chapter eight summarises the modelling scheme and presents a step by step guide. Some problems related to guidelines for modelling and analysis are also pointed out and put in the context of the thesis.

The thesis ends with some general remarks and some conclusions. The relevance of the work is discussed and some hints for future work are made.

After a reference list, an appendix is added. It describes the analysis of a feedback queue and is related to chapter seven.

Let us now, after this list of contents, look at the object of the study.

1.2 Object

Simulation is, according to Shannon (SHAN 75), an often quoted reference:

Simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behaviour of the system or of evaluating various strategies for the operation of the system.

This definition is very wide, and contains almost all possible applications of simulation. It is thus valid also for the kind of simulation that this work is restricted to: simulation of queueing structures by discrete event simulation. But even without this restriction much of general implications can be said. Let us look at that case first.

Simulation is, as Shannon quite correctly points out, to experiment with a model. There are, however, two important questions that have to be answered before this experimenting can take place: what should the model look like, and how should the experimenting be done. The first of these steps is usually referred to as modelling and the other as implementation. The latter is seemingly simple, and has over the years seduced many programmers to make them believe they know simulation.

A simulation model is usually implemented as a program in a computer. Before computers, with sufficient capacity for this kind of application, became generally available, it was common to simulate problems of continuous nature on analog computers, but to-day most of these problems as well are run on a digital computer.

Depending on whether the simulation is of discrete or continuous type the implementation will differ a great deal. It is therefore necessary to restrict the arguments to one of the two kinds. Since the main purpose of this work is to investigate the feasibility of simulation of queueing structures, and since these have a behaviour that can be described as essentially discrete, the presentation will be restricted to what will later be referred to as Discrete Event Simulation (DES). It is not obvious that this separation should be done, since especially the modelling steps of the two techniques have much in common, but both exemplification and terminology would suffer severely if the presentation would have been kept general. Simulation of continuous and discrete systems have developed along paths that have, at least so far, been heading in the same direction, but which have not crossed. It is likely, that a narrowing of them would make both prosper, but the efforts in that direction have not yet been successful. Recently, some progress has been made by a group round Zeigler and Ören (NATO 82).

There are many reasons why we want to use simulation. They have one thing in common in that we try to gain some information that for various reasons would be hard, or even impossible, to gain by other means. We could, for example, be interested in how a specific system, real or conceived, behaves under circumstances that are not easy to create in the system's natural environment. In such a case, simulation could be of help. In some other case, we could be interested in how some decision affects a system behaviour. Yet further examples could be picked from almost all human fields and activities, where simulation can be used to make something unknown intelligible.

In all these cases we have to be aware of the fact that what we are investigating is not the system itself, but a model, which of necessity is something that is simpler than the system itself. And since simplification means that something has been left out, it is not possible to gain the same types information from the model as from reality itself.

In spite of this, simulation is often the best we can do, but it is necessary to be fully aware of the limitations of the technique, and not push the conclusions too far. How far the results from a simulation can be extended, is mainly a question of how the modelling and the implementation are performed, and the lack of guidelines for these processes is a severe drawback of the technique. A trained and experienced practitioner can sometimes reach a high level of professionalism in the area, but since it

so far has been extremely hard to convey a good programming technique and a good feeling for modelling, trial and error has been the main way to learn to be a good simulationist. The interest in the theoretical and methodological basis for simulation has increased, in recent years, and some books and articles on the subject have been published (ZEIG 76, ZEIG 79, ÖREN 81, KREU 80). The problems are immense, however. The attempts to form some kind of general theory are heavily coloured by the background of the individual or group of individuals that has developed the theory. It can therefore be questioned, whether it at the present level is possible to form any general theory or foundation that can be of any assistance in practice. It is perhaps even unlikely that this could be done for a restricted field of application, or for a scientific discipline.

This does of course raise the question whether simulation, and the use of models as a whole, can be considered as science. There are many disciplines that have emerged from technology and engineering, that are in the same position. They have not existed long enough to form a well structured theory, and have in the past perhaps been too simple to have needed any. This is true for other disciplines as well, such as social sciences and psychology. It is of course not a coincidence that Kuhn (KUHN 62) picks most of his examples from natural sciences, and that Popper (POPP 80) deals mainly with physics.

It seems to be impossible for man to analyse himself, and this is probably true in a wider sense: it is impossible for an object to analyse itself. The possibilities to apply Kuhn's theory of paradigms to other disciplines than physics and related subjects, are limited by the dynamic behaviour of the studied phenomena, at least for the areas where man is a part of the system or has invented parts of it. It seems as if man himself, and the systems he creates, changes and adapt to new circumstances more rapidly than a common base for judgement can be formed. Kuhn has later acknowledged this view. Research in the field of technical inventions is made on systems that change so fast that the research community has no time to agree on any principles. Equipment grow old quickly and the only thing that remains intact over any substantial period of time is man's way of thinking, which leads us back to psychology.

There is a great difference between science and research, and it is of course possible to conduct research in an area that cannot be considered as scientific. It is in some cases possible to point out the fundamental principles that all scientists adhere to. This is not possible for the above mentioned disciplines nor for modelling and simulation however. The absence of these commonly accepted principles is presumably due to lack of need. Each practitioner has his own rules, and clings to them. But since computers entered the scene, things have changed rapidly. We have, firstly, experienced that the borders between technology and what perhaps could generally be referred to as "fine arts" have diminished, and, secondly, that the potentials are no longer limited by materials, supply and so forth, but by the capability of man to understand the complex behaviour of the things he tries to build.

If a foundation of simulation could be found, it would perhaps be possible to create a theory, that could be applied to a specific problem to set out the potential of a simulation analysis, and to tell whether the analy-

sis is worth-while or not. But since the paradigm of modelling has not yet been found (if it indeed exists) and since it, if it were found, would be likely to be vague and inconsistent, it is at the present level better to try to find some common properties within a restricted application area. The task becomes easier in such a case, but in no respect easy. The existence of successful simulation analyses has made people believe that the modelling process can be automated, and that the computer can do the hard work. This belief can be questioned. Before any attempt is made to automate the modelling process, it has to be stated what can possibly be gained from a simulation, and how reliable the results are. An automated process tends to hide these questions, and too much comfort is therefore often put in the results. Automation has to take place at a proper level, where the rules, guidelines or computer aid we give are sufficiently general to be applicable to a sufficiently large class of problems, but not so general that they introduce errors in the analysis or make the results useless. For a limited class of problems, this may be done, and this work is an attempt in that direction.

We have thus found, that the fundamentals of simulation are not well established, that it is not clear how a simulation model and program should be constructed, that we do not know how "true" a simulation is and that the diversity of applications makes it hard to give any more general guidelines. It is thus obvious, that research at the present level has to be directed towards an investigation of how simulation actually works, and towards the establishment of guidelines for restricted areas of application.

It must currently be considered ill-advised to work on some quasi-formal description of processes so vaguely and insufficiently described as modelling and simulation. Any description has to be just as imprecise as the knowledge it leans on, and modelling techniques have to be conveyed by examples and other similar means. This means that examples of successful simulation projects (and unsuccessful) are extremely important. The problem is however, that the importance of examples for the development of a more general theory is very rarely mentioned. Papers and books that report on simulation projects do usually end with a comparison between the simulation model and some measured data, and the more general implications, if there are any, are not observed or mentioned. This issue is closely related to the credibility of simulation results, since it deals with the generality of the results. If the merits and failures of a model were mentioned and related more often, a generality could perhaps be observed.

These issues will be addressed more thoroughly in subsequent chapters, especially when validation and credibility are discussed, but it is important to have them in mind throughout the text.

From this introduction it may seem that the simulation technique is of no value, and that the modelling is done at random and on intuition. This is to a certain extent true, but there are on the other hand no real alternative to simulation in many cases. The use of simulation has increased tremendously over the last ten years, and will probably increase even more in the future. The increased use and the impact simulation results have on important decisions, do of course call for more sophisticated methods and for careful treatment of models and results. Simulation

is in many ways a very powerful and efficient tool, but it has to be used with care. A critical survey is therefore more appropriate than an unreflected listing of results that show good correspondence with the so called reality. Let us therefore look at the modelling process a bit more closely.

2. Models

2.1 General

Models and the use of models are naturally associated with natural science and some abstract sciences. But they are used in all kind of contexts, and even in everyday life.

A model is a description of a phenomenon, real or conceived, that is made for us to understand the phenomenon better and/or easier. It transforms "reality" to a form that we are more used to and perhaps can handle with greater skill. Sometimes models are used to convey some insight. This is for example true for a literary metaphor. In this, the author tries to show a phenomenon in a new light, that brings out previously unknown or undiscovered behaviour. Such a model has that in common with a scientific model, that it tries to grasp the essential behaviour of the world around us. It transforms the behaviour into something that we are used to handle; it simplifies, generalises and enlightens. A scientific model does usually make use of some special kind of language or formalism. Most commonly used is of course the mathematical formalism, but other descriptions, such as graphs, symbols and computer languages, are also used. The mathematical formalism has a great advantage in that the rules for manipulation of a model description are known (with some notable exceptions), and that the result of the manipulation often can be checked to pertain certain properties that must have remained throughout the manipulations. A model that is described as a mathematical expression or as a set of such, does in some cases show some severe drawbacks, however. This will be discussed more in detail later, when analytical models are surveyed.

It is obvious that a model is built with a certain goal or for a certain purpose. The purpose is usually to describe a behaviour that is too complex to understand without simplification. But the simplification can of course be done in many ways, and it is not obvious which way is the "best".

What do we then mean when we say that a model is "good" or "bad"? A model is expected to mimic the world, or at least a part of it, as truly as possible. But before we can say anything about this, we have to know what we mean by "reality". Reality, or the real world, are frequently used expressions.

We hear them almost every day, and we know immediately what they refer to. The real world is "of course" that which we talk about, and which everybody has the same opinion about. We never discuss whether a chair is a chair or whether it really exists. In everyday life these things are self-evident and there is not much to argue about. But when we start to discuss how "true" a model is, i.e. how well it reflects some behaviour of a system that we want to investigate, then we have to be more careful.

Things that are seemingly self-evident may very well turn out to be immensely complicated when examined more carefully. As for a technical, economic or social system we may be convinced that it behaves in a specific manner and that the cause of the behaviour is as we think. A closer look may reveal behaviour and causes that totally change our system view. If a model is built on the initial knowledge and has to be changed, when the new behaviour is discovered, it is frequently observed that the model structure cannot cope with the new circumstances and that the model has to be rebuilt from scratch. If the model at this time already has been implemented as a computer program a lot of effort will be wasted and time lost. Consequently, there is a need for a modelling scheme that will force the modeller to observe the behaviour of a system beyond the initial knowledge or that received from a shallow investigation. In this light, many questions arise:

- 1) What do we actually want the model to describe. Do we want it to be "true" in all respects, or is it sufficient if it describes some properties of the system.
- 2) If we have two different models, how do we compare them. Can one model be said to be "better" than another. And if so, what standard of measure is used.
- 3) When a model is formed, what is it that makes us incorporate some behaviour while we exclude other.

The answers to these questions are unfortunately not clear. They depend on the background of the individual that forms, manipulates and utilises the model. The formulation of a model is usually preceded by some information acquisition, during which we try to understand the behaviour of the system and to identify its important properties. We filter, in this process, information according to our earlier background, but very little is known about this process. Some research has been made, but the results are still at a very basic level, and are currently of no use for explaining how models are built (BOUR 79). It seems, as if we are inclined to overlook behaviours that do not fit into the kind of model language we have at hand, and that this cripples our models. On the other hand we tend to include rather unimportant behaviour just because it fits our language. The history of the case study that follows later in the text will give an illustrative example of this. It can obviously be argued that this is due to incapacabilities of the modeller, but even if that is true, it is of interest to find out how a better modelling technique should be conveyed, so that such mistakes could be avoided.

To be able to answer the first question above, we have to split it a bit further. If we want a model to be true in all respects, we find that this is essentially equivalent to make an exact copy of the system, and hardly a model. We can therefore conclude, that a model can only be true in some respects. It is usually simple to tell in what respects the model ought to be "true", and it remains to grasp the pertinent behaviour (with respect to that specific property) and include it in the model. This is easier said than done, however. It is only for relatively limited areas that we can name important and unimportant properties. Some properties, that are of importance at one occasion are of small or no importance in some other case. What we need are some guidelines to judge the importance of

different system components. This can only be done for very limited application areas, and usually only after a complete specification of the aim of the study.

The second question above deals with the accuracy and fitness of a model. Two models can, with respect to a specific parameter or performance measure, be almost equally accurate. If one of the two models contains properties that do not have any impact on the behaviour we are studying, they can be excluded without loss of accuracy. This statement may seem to be much of a truism, but it is not uncommon that models are said to be accurate just because they are detailed. If the details are unimportant with respect to the investigated behaviour, it is obvious that the models carry a lot of dead weight.

If two models are to be compared, there must be a way of comparing them. This implies some kind of standard of measure. Zeigler (ZEIG 76) discusses the use of an experimental frame to define the set of measures to be investigated. But this does not answer the question of how to compare models in a more general sense. In a general case, the problem is too complex. Models can be accurate in many different respects, and a measure of accuracy that could apply to all kinds of models would probably be of little importance and use. But for restricted application areas, measures of accuracy and relative importance could be suggested, and such will be given in what follows.

The third question is perhaps even harder than the two previous. It does, to a greater extent, depend on the individual who includes or excludes the detailed behaviour of a system. This process seems to be almost impossible to describe, and even harder to give any rules for. People "believe" properties to be of importance, and it is usually only experience that can make them change their view. And most of the time it is impossible to know in advance what to remember and what to forget. A detail, that at first seems to be of no importance at all, may at a later stage be seen to have significant impact on the system behaviour. But if the modeller does not even remember that detail, it is not easy to tell how he should have been able to incorporate it in the model. The process could perhaps be improved somewhat by a guide to data acquisition, but this step is usually performed at the end of the modelling process, and is therefore biased by the model. A scheme that forces the modeller to acquire almost perfect knowledge of the studied system at a very early stage is certainly the best way to secure a minimum of surprises later on. But how does one know when the knowledge is as good as it can possibly be? For simulation of queueing structures, a tentative proposal will be given.

But even if the three questions above could be answered satisfactory, problems still remain. If we look into the behaviour of a system, and find that it under certain circumstances behaves in a predictable manner, we do also find that circumstances change and make our predictions vague and imprecise. We can also find that it in most cases is impossible to describe, and certainly to model, all the different behaviours we encounter. We would say that the system behaves randomly. This would of course be untrue, since everything has a cause, even if we do not know it. Different people have different opinion about the randomness of phenomena that we experience. Some would say that our tax system be-

haves randomly, while others, though not very many, would say that it behaves as intended. This difference in judgement is of vital importance to the goodness of models. To be able to use stochastic models in a proper way is perhaps one of the most significant capabilities of a good modeller.

Since behaviour has a cause, it is, formally speaking, always possible to describe how a system will react under given circumstances. But in all practical situations, this is not feasible. The point, at which a system could be regarded as random, is one of the crucial decisions to be taken in the development of a simulation model. In essence, it is a question of information. If the information we have about the system is unlimited, then it is formally possible to build a "perfect" model. But since this hardly ever is the case, we have to stop somewhere. But where? Whatever the choice, it sets a limit to the accuracy of the final simulation model. When we introduce some stochastic behaviour we give away some information about the system.

A model is something of a transformation, by which we transform "reality" and show it in a new shape. This transformation can be described by the words "as if". "Reality" behaves as if it was the system the model is a perfect description of. This "as if" involves indeed a generalisation. In order to form a behaviour that is simple enough for us to understand, we have to simplify. Simplification is very much the same as generalisation. From a fairly limited knowledge we say that we believe that the system in essence can be described by the model. Such simplifications are very common in everyday life, and are perhaps the only way we have to gain knowledge. Although we do know that the world does not always behave quite as we say, we argue that it generally, or usually, behaves as we say it does. We are sometimes accused of doing too simple generalisations, when we for example say that all Swedes are blond and silent. But if we do not generalise, we are often not able to say anything, at all.

In a modelling process, generalisation comes in when a behaviour, that is too complex to be described exactly, is to be simplified. We then have to say that the behaviour of that system or component in general terms and under normal circumstances is so and so. The generalisations have great impact on the validity of the model, since they seriously limit the circumstances under which our model is valid. The circumstances, under which our model is valid, are sometimes referred to as the experimental frame of that model. This terminology has emanated from the area of modelling and simulation of continuous systems, however, and does not seem to be especially appropriate for discrete systems. Simulation of continuous systems belongs more properly to the group of analytical models, since analytic tools are used to describe the properties of the system, and simulation is used to solve the often very complicated mathematical expressions that arise. The possible experimental frames are here set by the mathematical methods and techniques that are used. In a discrete simulation model of a system it is harder to speak about an experimental frame, since it is virtually unlimited how well the target system can be described. No mathematical technique limits the potential possibility to model a system as accurately as necessary.

There are two main groups of modelling techniques: analytic models and simulation models. These are investigated below.

2.2 Analytic Models

An analytic model is a model in which the properties of a system are described by mathematical expressions that can be manipulated with mathematical methods. Mathematical expressions and methods are of course used in other models as well, but the behaviour of the system is not necessarily expressed by mathematical relations. A combination of simulation and analytical models is usually called a hybrid simulation, but the classification is not strict and it is hard to tell when an analytical model ends and a simulation starts, especially if the solution of the analytical model involves numerical techniques. Hybrid simulations have been discussed in various papers lately, and will here be discussed in connection with analysis of queueing structures more generally.

Formally, it is quite possible to describe a very large class of problems and systems by analytical models, but their usefulness are restricted by the possibilities to solve the resulting equations. Formerly, only fairly simple equations were possible to solve, but the advent of computers has enlarged the group of tractable problems significantly. The new solution techniques have introduced some numerical problems, that occurred rarely with the closed form solutions that resulted from the "old" techniques. This has somewhat narrowed the gap between simulation results and results obtained from analytical models. Simulation models and analytical models differ in how the stochastic assumptions are implemented and in how the results are interpreted. To be able to use an analytical model one has to transform "reality" into mathematical expressions that thereafter are manipulated exactly. The results appear as numbers that are exact for that model. It is therefore formally possible to compute the probability for an extremely unlikely event in the model. It may be that this probability says very little about the likelihood for that event to happen in the "real" system, but that has to be blamed on the assumptions, not on the computation. With simulation it is somewhat the other way around. The assumptions made in simulations are usually less severe, but the results appear only within certain limits and cannot be considered as exact, not even for the model. The question is when the accuracy of the results is of the same magnitude as the accuracy of the assumptions. Is it for example with the assumption of some service times distribution meaningful to calculate a probability which is around ten raised to minus ten? And is it, in the same way, sensible to run a simulation model for weeks to estimate a similar probability?

Analytical models have the advantage, among others, that they give results with substantially less computational effort than simulations. This is one reason why an analytical model is to prefer to a simulation model. But it is generally agreed that simulation models can be made more accurate (which can sometimes be questioned), and simulation models are therefore used to check the accuracy of analytic models. But when results are compared, they often differ. It is not uncommon, that the differences between simulated results and analytic results have other causes than statistical insignificance.

Analytical models do generally require more crude assumptions than simulation models, but can very well be used to make a preliminary in-

vestigation of a system to find out basic properties. The use of analytic models for this purpose will be shown later. Analytic models will also be discussed in conjunction with queueing structures.

2.3 Simulation models

2.3.1 The simulation technique

As stated already in section 1.2, simulation is to conduct experiments with a model. Once the model has been made, it is implemented as a computer program. Everybody who works with simulations would agree to this definition, but would after this make different descriptions of what simulation actually is. It is therefore necessary to specify the use of the word simulation. Simulation is most commonly meant to involve modelling as well as implementation of the computer program.

It is very natural to make a division of the simulation technique based on the behaviour of the studied system and how it can be described. If the laws that govern the system change is complicated, for example not simple functions of time, but given by differential equations, simulation means to evaluate numerically the system behaviour under these constraints. The difficulties to simulate such systems are due to the complexity of the mathematical relations that the differential equations express. It would formally be possible to solve the equations analytically or by numerical methods of simpler kind than simulation, but this may be unfeasible or inconvenient. Simulations of this kind are called continuous simulations and are not dealt with here.

The other type of simulation is when the model changes its state abruptly, or when the change is so rapid that it can be considered instantaneous. The system is, between two consecutive changes, in a well defined state. The state is in a simulation described by the variables of the program. The variables may not be defined or directly accessible by the user, but are all the same the memory of the system. A change of state is called an event, and simulation of events that occur at discrete points is called Discrete Event Simulation (DES).

Simulation is for ad hoc purposes defined as:

Simulation is a modelling and implementation scheme, in which the behaviour of a system is imitated. The system is assumed to alter between well-defined states, and the statespace is enlarged until an appropriate level of detail is reached. The ambiguity that remains at this point is modelled as intervals between change of states which are governed by stochastic distributions. After the model is constructed, and implemented as a computer program, experiments are conducted and measurements performed.

This definition may at first seem to be limited, but quite a few systems can indeed be described as altering between well-defined states with certain intervals. The class is actually so large, that it has to be limited

even further to be handleable. But the definition makes it, even without the limitations that will be needed later, possible to fix some crucial points in the modelling process.

- a) What is meant by a proper level of detail, and when is it reached.
- b) What kind of stochastic distributions should be used to model time delays.
- c) When does the simulation end and the outside world begin.

These questions will be addressed for the limited application of queueing structures. They are of course relevant for other areas as well, but cannot be answered generally. The answers depend on the implementation technique, the structure of the problem or system and on desired accuracy. These presuppositions have to be formulated before any analysis can be made. This will be done in chapter three. But before that, a brief look at the different approaches to modelling and implementation.

2.3.2 Different approaches

When we are modelling, we need some kind of support to help us describe and organise our thoughts. This is often referred to as different approaches, and the three main approaches to Discrete Event Simulation are:

Process Approach

Event Approach

Activity Approach

The text will now for a while be somewhat more concrete and detailed, but it is impossible to discuss the different approaches without getting into detail. The Event Approach and the Process Approach are more closely related to each other than to the Activity Approach. The main difference can be expressed in terms of the system view. The Event and the Process approaches look at the system from inside, so to speak, while the Activity Approach applies more of a bird's-eye view. This means, that the Activity Approach demands the user to have a more complete knowledge of the system behaviour, which can be quite impossible to meet if the system is complex. An Event or Process Approach makes it possible for a modeller to concentrate on an isolated part of the system if he knows the impact the other components have on the part he tries to model. The Activity Approach, on the other hand, treats all events at the same time, and the user has to specify the next possible events that can follow on the one that has just occurred. He can then choose from a set that contains all possible events in the system. This number can for more complex systems can be quite large. Depending on the type of event, different actions are taken, appropriate for that specific event. The actions that take place concern different system components and are therefore usually of diverse kinds. The Process Approach puts more emphasis on the behaviour of the components involved in the event. A more important difference among the approaches is how the technique is

implemented. The Event Approach is supported mainly by SIMSCRIPT (KIVI 73), which does not enable the user to build any structured programs, since the language is based on FORTRAN and has poor facilities what data structures are concerned. The Process Approach is supported by GPSS and SIMULA (SCHR 74, BIRT 73), but GPSS has the same deficiencies as SIMSCRIPT as regards data structures. GPSS is an excellent language for small and medium sized problems, but for large and complex systems, which are the main concern in this text, it gives too complex and unstructured programs. The conservatism among practitioners is great however, and it is not easy to spread a new language and new modelling principles. Simulation conferences are used poorly in this respect, and the willingness among participants to discuss these matters is very limited. Most people tend to speak in favour of "their" language, no matter what new achievements have been made since they started to use a specific language decades ago. Some attempts to improve the event and activity approaches have been made in recent years (EVAN 81), but these have mainly dealt with adjustments or new facilities, made possible by increased computer capabilities. They have not to any significant extent contributed to the methodology of simulation.

Since all three approaches describe the same behaviour, they have much in common. But they do all the same represent different ways to think about simulation, and are therefore hard to treat at the same time. They are supported by dedicated simulation languages and the terminology is biased or at least influenced by the support language. This is in no way bad, since we do need concepts to express properties of the systems we study. But some languages have concepts that are at too low a level, and these are usually not general enough to be able to express relevant properties of more complex systems without getting unnecessary detailed. The language that currently contains the most advanced concepts is SIMULA (BIRT 73). It is by no means a perfect language (if such a one exists (KREU 80)), but perhaps the best we can do at the moment. SIMULA implements the Process Approach, and contains language elements that simplify scheduling of events and handling of queues and lists. Apart from this it is a versatile general purpose language, and it can be used for programming of various tasks, not only simulation. The concepts that are implemented for simulation are sometimes a bit too general, which leave the programmer with too many options and make the programs error prone. But as a base for future development it is quite adequate, and it will be used here to enlighten some fundamental principles.

A simulation program is made up of many parts. There are input-output, initiation, description of system components and so forth. When the program is executed, only some parts contribute to the major part of the execution time. Input and output, initiation and similar procedures are usually executed only once, and most of the time is spent in some of the procedures that describe the system behaviour. These components are the processes of the Process Approach. Independent of the application, they have a very similar structure. A process consists of a closed loop (closed in the sense that it usually runs as long as the simulation lasts), that is broken up in smaller pieces, each ending with a scheduling statement. The number of scheduling statements within a loop is related to

the complexity and the accuracy of the system and the model. The time delay that is associated with the scheduling plays an important part in the modelling process, as will be shown later.

The most common way to define a process, is to model a component of the system. Such a component can for example be a machine, a service facility, a human being or any other welldefined part of a system. In these cases it is usually quite clear how the system should be partitioned and what processes should be defined and modelled. There is another way to define a process however, but not as commonly used. A process could of course also be defined by the flow of some entity through a system. In such a case the process is related to the entity and not to the components the entity visits or utilises as it propagates through the system (FRAN 77). This view is related more to a functional description of the system than to a physical. It could, as an alternative, have been used in the case study that will be described later on, and will be discussed in this context.

The Process Approach has, usually with SIMULA as the fundamental component, been elaborated and used in new approaches (FRAN 77, BIRT 81). These more elaborated forms contain elements that simplify some steps in the modelling process, such as data collection and statistical analysis.

The development of a simulation program is a complex process that is difficult to describe in general terms. Different modellers and programmers have different ways to tackle a problem and the process from the first contact with a target system to a final program is greatly influenced by this. It is however, in spite of these individual dissimilarities, possible to find some relevant questions that have to be answered before a model can be set up and before a simulation program can be constructed. These questions are general however and to be aware of them does not to any larger extent help the modeller or the programmer in their specific tasks. But they can be of help when we try to find out how a model is formed. An extremely important issue, which will not be dealt with here, is how to describe the model at the first tentative stages. Common programming languages are usually too detailed at this stage, and some hybrid, using both programming language concepts and ordinary prose, is probably the most versatile method.

It is likely, that the first tentative contact with an unknown problem is the most crucial in many respects (at least what modelling is concerned). At this stage we try to form some kind of opinion about the system that is studied. We do rarely start from scratch, but are usually informed to some extent, albeit not sufficiently. And even if we do not know anything in particular about the system, we do often have an opinion about the context of the system or about the environment that will influence the system. The information we gain will at any rate be filtered by our opinion and by how we judge the importance of the information we receive. This may make us overlook behaviour that is important. But since we could not possibly be aware of the situations when this happen, there is not much we can do about it. The effect is in spite of this of great importance.

An experienced simulationist or modeller does usually have a great ability to extract pertinent information from documents, well-informed

sources and by experiments and measurements. This information acquisition is not unbiased however, and the principal and most common error is probably that the feasibility of the simulation model influences the possibility to at a later stage retrieve the information. It is likely, that our brain, without us being aware of it, already when we receive some information goes through some of the modelling and programming steps and decides that the information is too complex to be incorporated. This implies that any scheme that should be a guide in a modelling process, has to force the user to answer questions that will make it hard to get by pertinent information. Other properties, besides this, have to be expected from such a scheme, but this one is necessary. It is likely that errors of this kind are hard, or even impossible, to discover without a rather detailed formulation of the model. The inability to observe incompletenesses in the model at an early stage does usually lead to severe interrupts in the schedule, since the level where the deficiencies are discovered is: "formulation of the program". And it is well known that once this level is entered, the pace is decreased and the ease with which corrections can be made is much lessened. It is therefore an important demand to put on a modelling scheme that it should allow the user to do as much work as possible as "paperwork" without any programming. The "paperwork" could of course be aided by a computer, but should not involve any detailed programming. A tentative formulation of such a scheme will be given in chapter eight.

The environment or context of the system does usually have a great impact both on the modelling and on the behaviour of the resulting program. Very few "real" systems can be isolated from the environment. Most systems interact one way or another with the world around them, and this interaction has to be considered in a simulation model. It is of course a subjective opinion where the system ends and where the "outer world" begins, but regardless of this, it is necessary to represent the interaction in some way. There is a difference between modelling the system and modelling the context, or, to be more precise, the impact of the context.

It is, under normal circumstances, quite simple to increase the level of detail. Generalised and simplified behaviour can be broken up in smaller pieces which are copied more precisely from "reality". Such a process does usually lead to a more accurate model (even if this is not always true, as will be discussed later), and it is in most cases possible to define a level at which the model is sufficiently accurate. Stochastic distributions that are used in the simulation do in such a case get more and more definite and represent behaviour that is less and less ambiguous. Such a description is not necessarily true if we move away from the system. We may, as we try to increase the detail with which we have represented the environment, enter areas where it is harder to define and grasp the behaviour of the components that make up the total impact. Take for example a telephone exchange. It can be shown that arrivals to the exchange, as calls, form a Poisson process with extremely good accuracy. If thus the environment of a telephone exchange is represented by a component that enters calls into the exchange as a Poisson process, we have a good representation of the environment. But if we try to increase the accuracy of the environmental representation by dividing it into

other processes, such as an individual call process for each subscriber, we do not necessarily end up with a "better" model, since we do not know how individual subscribers behave. It is therefore clear, that one of the most crucial decisions to make in a simulation study is where to put the border between the system and the world around it. It also follows that the simulation does not take into account the impact of the system on the outer world.

Since the model does not interact with the environment, and since the behaviour of this is independent of the behaviour of the model, there will be a special representation of the components in the simulation that represent the environment. These components will influence the simulation and they are preferably called Generators. A Generator will generally consist of a closed loop in which a scheduling statement and some activity that influences the system behaviour will be incorporated.

The environment can consist of many interacting parts which produce a variety of behaviours that from the viewpoint of the system look quite random. It is therefore in general necessary to describe the delays in the closed loop as some stochastic distribution. A Discrete Event Simulation is often viewed as a sequential treatment of some sort of entities that flow through the system. With this view it is possible to describe the generators as components that enter the entities into the system. Referring to the previous example, it is obvious that it is calls that enter the telephone exchange, the question only being where they are generated and with which intervals.

The selection of Generators is closely related to measurement of input data and to results expected from the simulation model. But these issues are also related to the accuracy of simulation studies, and are therefore postponed until these matters have been discussed.

2.4 Accuracy of models

The accuracy of models is disputed in the scientific world and elsewhere almost constantly. Every now and then a big row starts about a model, the assumptions it is based on and the conclusions it brings forth. This can easily be observed in economics, where people often accuse each other of too simplified models and views. A recent example is the controversy about the optimal point of taxation. The argument of primarily Laffer, is that since a tax rate of zero percent, as well as a tax rate of 100 percent, will give the government no revenue, there must exist a point somewhere in-between that maximises revenue. The argument would probably not have been disputed had it not been used to argue that the tax imposed by many governments is already too high and that a reduction would in fact increase revenue. A tax revolt in California in the late seventies is also used to prove the thesis. In this revolt, taxes were actually decreased and some investigations showed signs that worked in favour of the idea. Objections from people who had to defend the tax system in use and who had to disappoint people whose expectations on a

tax reduction had increased by the arguments of Laffer, were of course very strong. Recently Gardner (GARD 81) had a most interesting article in *Scientific American* where he showed the hollowness of Laffer's arguments. Gardner looks at some of the economic models that have prevailed during different time periods since Keynes. He points out that economic theories are often presented as curves, extremely simple curves, and that economy sooner or later behaves in a way that violates the curves. In the 1960's, it was generally agreed that inflation and unemployment worked in opposite directions so that high unemployment was combined with low inflation and vice versa. But then something peculiar happened. The economies of the West got into a state of stagflation, where inflation and unemployment raised simultaneously. The Phillips curve could thus not be used to explain how the economy worked but had to be replaced by something else that could incorporate stagflation. It is beyond the scope of this text to deal with the new theories, but such have appeared and they are better in the respect of stagflation but they have other deficiencies.

Gardner has examined the Laffer curve and shows that it is an very simplified model of the relation between tax rate and revenue. He calls it an example of "beauty bare", and says that this is how we want theories to be. Reality however, behaves not quite as smoothly as the curves. Gardner has devised what he calls the neo-Laffer curve. It is based on statistics for the U.S. economy over the last 50 years and resembles the Laffer curve at the beginning and end points. But in the interesting region, i.e. at realistic tax rates, it gets into what Gardner calls the technosnarl. The curve is here distorted and there is almost an infinite number of points that could be picked to prove or disprove the thesis of Laffer. There are for example several points that maximises the revenue, not only one.

This example is interesting in many respects. It shows that extremely simple models sometimes can be widely accepted, that it is easy to put forth models that are hard to check the validity of and that models at times are accepted because they talk in favour of something desirable. The accuracy of Laffer's model is quite obviously not of the magnitude needed in economics, but it has served as a conceptual model for an issue that is most relevant and which has great impact on the way in which taxation is handled. It also shows that models not always have to be accurate to be of use. Even simple models can bring forth information that is quite sufficient at the moment. But it is also obvious that one has to be extremely careful not to make extensive conclusions based on simple models. The Laffer curve can serve as an illustration of the problem of a proper level of taxation, but it cannot be used to argue that the tax rate should be changed in any specific direction. For this, the model is too simple. The example can also serve as an illustration of how general assumptions severely limit the accuracy of the model.

Models have to be built in accordance with the demands, and made sufficiently but not unnecessarily accurate. But before such a goal can be accomplished, accuracy has to be defined and possible to measure. In a general case, neither can be done, but for simulation models of limited subject areas, the issue can be discussed and some fairly general implications found.

What is then actually meant by the accuracy of a model? Generally speaking it is a measure of how "true" the model is compared with "reality". But such a statement involves vague terms as "true" and "reality" and cannot really be said to shed any new light on the meaning of accuracy. To be able to state anything more substantial about accuracy of models, it is necessary to look at the process of modelling and programming as well as measurement a bit closer.

Without addressing the question of whether man can have any true knowledge about the world around him, it can be said that the opinions we form about the world and phenomena in it, are based on experiences we gain through our sense organs and the information processing done in our brain. The images created by these processes may of course in a philosophical sense be imaginary, but without this starting point any further discussion would be useless. Let us therefore assume that we have some correspondence between our opinion about the world and the world "as such". The important thing in this context is not that the opinion may be imaginary, but that it is not complete. If we study a system and try to gain as much information about it as possible, there will always remain some rest that could be observed if we looked at the system from some other standpoint.

Look at the following picture. It describes – in one way, there are of course many other ways – the different steps in a modelling and measurement process.

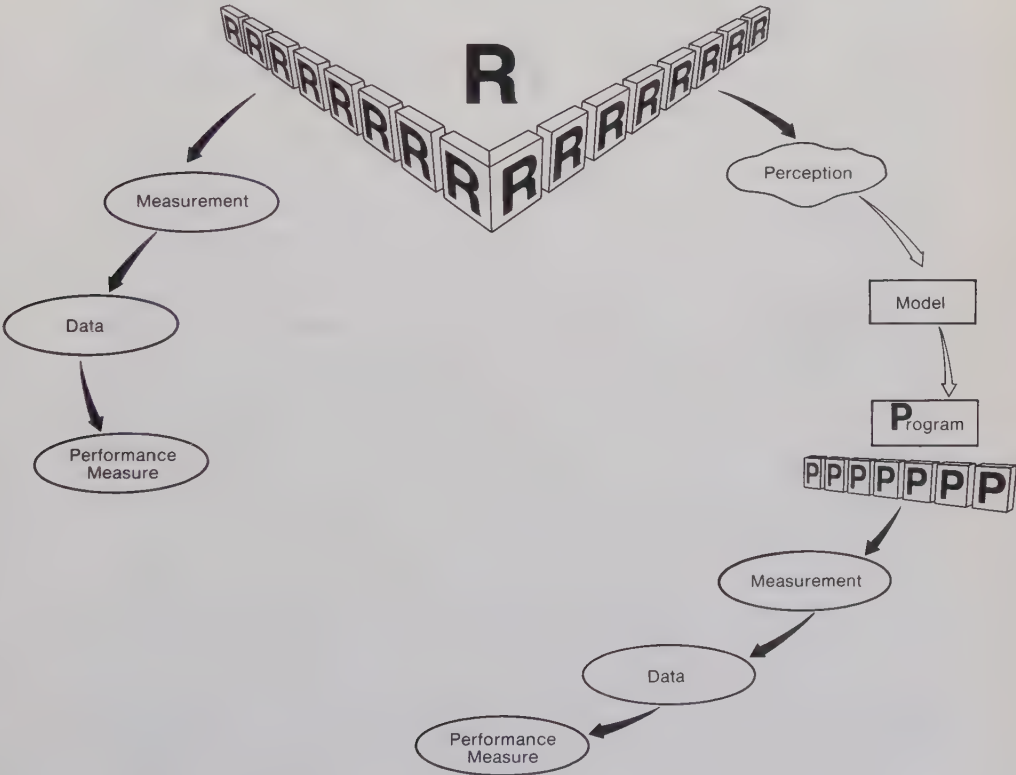


Fig. 2.1 Modelling and measurement processes

Let us, for the sake of clarity, investigate the right and left branches of the picture in turn.

At the top of the picture "reality" resides. It is essentially impossible to seize, but we form some kind of opinion about "reality". Let us call this opinion a perception. The perception can also be deduced from already gained information and does thus not necessarily have to be directly observed. There are for example entirely conceived systems that do not exist in the sense used here. But these conceptions are of course based on earlier experiences and put together from similar systems in the "real world". The images that constitute our perception can be viewed as pictures taken of the system when it is in one of many possible states. This way of looking at the system is most profitable when it comes to explaining what kind of information that is obtainable from a simulation.

Our brain has the ability to average or concatenate pictures taken at different times and from different viewpoints. From this multitude of pictures an opinion is formed, but it is currently not known how this process works. It is anyhow important to observe that we can increase our knowledge of a system by watching it over and over again and at certain times observe something novel that gives some unexpected insight. This possibility is sometimes transferred to simulation studies as well, but is perhaps not as useful in that case.

Information floods around us constantly, and we filter it and bring in what we are interested in. If we are to build a model, we have to behave somewhat differently. In such a case we are aware of the risk of omitting pertinent information and we therefore try to get as well informed as possible before we start. We do however, at some stage, consider ourselves ready and embark on the modelling.

If the system that should be modelled is fairly simple it is possible to form the model as an alternative perception and to keep this picture in mind. From this picture it is then possible to construct a program, simulation or calculation. But if the system is more complex, it becomes hard to keep the model in mind without any written or drawn support. The tools available for such support are very limited, and the most important future research objective in the simulation and modelling area will be to invent and develop a versatile modelling language. Some tentative efforts have been made (STAW 82, KREU 80), but much is still to be done. Let us here assume that the model is possible to describe in some way and that this description can be used as a base for program development. Let us also assume that the intermediate steps in the chain are performed without error and that this can be checked. How this should be accomplished is treated in chapter four.

After the modelling, a program is constructed and tested. In this program, measuring routines are implemented that perform activities similar to those of the perception process. The similarity is restricted however. The dynamic and adaptive structure of our brain enables us to compare and form judgements of phenomena that are hard or even impossible to measure in a simulation program. This inability of simulation has been the cause of a great deal of frustration and unnecessary modelling over the years and will be discussed more thoroughly later. Generally, only entities expressible as numbers can be measured and compared in simulations. One way or another data are collected and treated accord-

ing to some statistical routines. These statistical routines can be more or less advanced, and give the measured quantities more or less statistical significance. For the present discussion it is enough to observe, however, that the result of the measurement and the statistical treatment is a number with a certain significance.

It is this number that should be compared with some other number, obtained by other means, to assert the accuracy of the simulation model. But before any comparison can take place, the other branch in the picture above has to be investigated.

Already when the first idea about an investigation of a system is formed, there exists an idea about the result of the investigation. In fact, this is a step that should be carefully analysed before any simulation model is built, since it in this process may be observed that the interesting quantities cannot be investigated by simulation. But if there is a strong belief that the simulation can be successful, quantities are formed already at the perception level and which express some property of the system. This opinion is then used to formulate measurements that should be performed on the system. Measurements are made in approximately the same way as in the simulation program and data are treated with similar statistical methods. The result of the measurement is a number which could be compared with the number obtained from the simulation. If they agree it could be an indication of an accurate simulation model. It is important to observe however, that agreement is a necessary but not sufficient condition for an accurate model. Simulated and measured data may very well agree without an especially accurate model. The model can be sufficient for the defined experimental frame, but may not be useful in some other, and perhaps larger, frame.

That much for the moment about accuracy if the measurements do agree. What if they do not?

As can be seen from the discussion above, there are many steps involved in the production of simulated data and the numbers deduced from these. This is true for data from both the simulation and from measurements on the "real" system. (There is another aspect when the "real" system does not exist, i.e. when the simulation is a picture of a conceived system. Simulations like these have their own problems which will be dealt with later). If we look at the right branch first we find a number of individual steps that can contribute to the disagreement. Some of them are listed below:

The information about the system can be wrong or incomplete. We may have misunderstood the descriptions or misinterpreted the direct information. Already gained opinions may have biased and twisted processing of the new information. A number of things may have caused us to get a wrong impression of the system we wanted to investigate. Such errors are of course fatal, and their effects cannot be repaired later in the modelling chain.

When we build the model we must, as described in the previous sections, make simplifications and generalisations. These limitations of our knowledge are not equally pertinent and we may have done the wrong types. The modelling does also involve insertion of additional assumptions, like for example selection of stochastic distributions. Considering the avail-

able information, more than one distribution may fit, and different picks may give different results when the program is run.

The model is transformed into a program, written in some standard programming language or some dedicated simulation language. This mapping may obviously be erroneous. This is probably the most common error, but fortunately also one of the easiest to discover and correct. It is possible however, to have errors in a program that produces seemingly correct data. To get rid of such errors is a part of the verification process, and will be dealt with later, in chapter four. With a good verification and cross-checks it should be possible to almost eliminate programming errors.

Data produced by the program is usually stored on a file for later processing. There are abundant literature about the statistical methods that can be used to investigate the statistical properties of simulation data. The available methods are at the present level in most cases sufficiently accurate compared with the accuracy of the other steps in the simulation process.

There are internal substeps of each of these steps and the modelling for example is not a well-defined process. These internal steps will be treated separately.

The left branch of the picture is somewhat simpler. But even here we have steps that can contribute to disagreement of results. Some examples:

The appearances of the measurement routines are not independent of the perception we have of the system. We have an idea about processes present in the system and we try to measure their behaviour. But there is a profound difference between measurements performed on the "real" system and measurements performed in a simulation program. The behaviour of the simulation program is to some extent known in advance, while the behaviour of "reality" can be surprising to us. This means for example that it can be quite informative to measure a sequence of events in "reality" to see if such a sequence can appear, while it is less informative to do so in a simulation.

Let us look at some of the steps in the left branch:

The measurements can be based on a wrong impression of the system behaviour. These errors do not necessarily have to be the same as the ones that made the simulation model erroneous since it may be some other kind of information that is needed to construct the measurement. The routines can therefore measure one thing while we believe them to measure something else. The error can be more or less grave. It can either be that the measured quantity actually is what we are looking for, but that it is biased or that the circumstances under which the measurement is performed is dissimilar to the one performed in the simulation program; or we may be measuring something quite different from what we believe.

Data produced by measurements on "reality" is usually stochastic in nature, and must therefore be treated with the same kind of statistical methods as data from simulations. Obviously the same kind of argument apply here as in the right branch. But the analysis of data from measurements on "real" systems is harder in that the background pro-

cesses are not known in detail, if at all. This leaves room for all kinds of interpretations and data from such measurements has to be carefully examined before any general conclusions are made.

But even this fairly long list of possible errors is far from complete. A few more examples could be given. As already stated, there is a strong interaction between the perception of the system and the formulation of the measuring routines. The problem is however, that it is often not the same individual who builds the model and who performs the measurements. The modeller is often supplied with data measured by someone else, and data is perhaps collected in a way that differs from that used in the simulation program. The encountered differences can usually be related to the different perceptions gained by different individuals. The only possibility to get around these errors is to carefully specify the circumstances under which measurements should be performed. Such specifications are not easily given in a general case and a more useful discussion could be made for queueing structures. The discussion is therefore delayed until these matters have been dealt with.

If it would be possible to form some kind of standard of measure of the accuracy of a simulation result, this would of course have contributions from all the steps discussed above. It seems however that the only step in which this could be accomplished is the statistical analysis. Much have over the years been done in this field and the knowledge collected exceeds by far the corresponding knowledge about models and model development. Since a general standard of measure for simulation model accuracy is not likely to be found, there is no alternative but to create an awareness among modellers about the possible sources of error and simplifications that are present in a modelling process. The present text is an aim in that direction.

Another reason for a greater awareness among modellers is that if it existed, much more of a simulation study could be done as paperwork, and the time prospect could be substantially shortened. If for example the possible outcome of a simulation was thoroughly investigated in advance, it could perhaps already at this stage be stated that the unknown properties that the simulation was supposed to give information about, could not be investigated by simulation. It is a pity that so few unsuccessful simulation projects are reported on and that very little time at conferences and space in journals are devoted to discussions about the limitations of the simulation technique. But this will probably come as the number of decisions based on incomplete and erroneous simulations increases. Before we proceed, let us limit the objective and define queueing structures.

3. Preliminaries and presuppositions

3.1 Queueing structures

3.1.1 General

Queueing is a more common activity than we perhaps assume when it is described as queueing theory, queueing networks, priority queueing and so forth. Commonplaces like shopping, go to the bank and similar things do indeed involve queueing at times, and many other activities that we do almost every day, as to use the telephone and drive on the highway, do also mean that queues are formed. Sometimes when queues are formed, they cause problems or at least annoyance. We do consequently try to get rid of them at times or to minimise their effects. If we could accomplish that, many things in our world would be easier and we could make many systems cheaper and and more versatile.

It is therefore not a severe restriction to limit the investigation of the potentials of the simulation technique to simulation of queueing structures. The state of the art of the simulation technique requires such limitations for any successful investigation of its potentials. There are of course other sorts of limitations, inspired by different applications of the technique, that could be used to investigate the possible advantages of simulation as a tool for analysis of such systems. This requires a substantial insight into the specific problems and demands of different applications as well as for that application defensible limitations. The applications of queueing structures are wide and they are used for analysis of telecommunication and computer systems. In these, time is a most relevant parameter, and queueing structures could be used to estimate the delay an activity will encounter in the system.

Most systems, not only technical, work under some kind of constraint. They may for example not be allowed to cause delays longer than some fixed number, to get overloaded or malfunction too often. The restrictions may in some systems never be violated, as in a nuclear power plant where temperatures are not allowed to raise above a certain level, ever. Sometimes we require the restrictions not to be violated too often depending on our demands. For example in the telephone system, in which we do not want to be blocked "too" often. To be able to answer questions about these events, we have to have a technique to model the system and perform experiments with that model. The two main techniques available for queueing structures are simulation and analytic methods. These can be used to estimate a number of relevant properties, but have their limitations, as all methods. Simulation is the most dynamic of the two, but it does usually take more time to implement and run, which is a severe drawback in most cases. There are others, but these will be discussed later in the text.

There exists a number of theories for analysing queueing structures. These theories and simulation should be seen as complements to each other. Their advantages and drawbacks are not of the same kind and they could therefore be used simultaneously to obtain different properties of a system. Let us investigate their relative merits in turn, but before that something about the concepts used in queueing structures and how these are used to describe them.

3.1.2 Queueing structure concepts

Queueing structures is not a homogeneous concept. It is therefore difficult to recommend some standard publication to the novice. Much can be learned from queueing theory, but some of it is quite cumbersome to understand. And in addition, most of the theories are not needed for simulation. There are structures of queues that are not easily described as mathematical problems, which practically all queueing theory problems are, and it may at times be a restraint to be too influenced by the notation of queueing theory. But a combination of queueing symbols and verbal descriptions is usually quite sufficient. The basic elements of a queueing structure are a server and a queue, and they are usually linked together in the following way:



Fig. 3.1 A simple queueing system

This basic structure can be elaborated to contain many queues and/or servers, but the basic properties are described by this symbol. A queue and a server do not usually work alone but are interconnected to form a network of queues, as this:

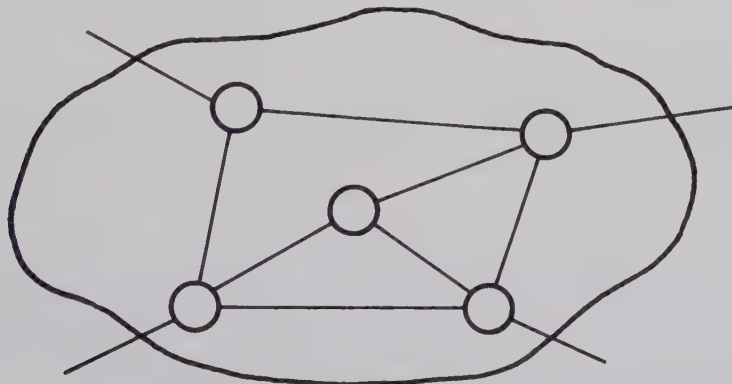


Fig. 3.2 A queueing structure

Such a picture shows the topology of the structure, but in addition, there are a number of properties that have to be specified before we have enough knowledge about the structure.

There must be some kind of entities that flow through the system, and these are usually called customers or jobs. The customers demand some kind of service from the server and we have to specify the amount required. In queueing theory this is often done by a stochastic distribution. Such a constraint is not necessary in a simulation though it is often imposed. Besides this, we may specify other things, for example: how customers should be routed, i.e. how they should be moved from one server to another; if there exists any dependencies in the system, i.e. if the servicetime in one server depends on servicetime in some other, on status of the system or on some other system parameter; if some customers have priority over other customers; and a number of similar constraints that enable us to deduce the system behaviour and to build a model. In a more general meaning we have already at this stage modelled the system to a certain extent, but we have not yet thought about the interesting properties of the system or about the implementation.

An important question, that has to be answered at this stage, is what we are looking for, and why. It was said earlier that queues caused delay and that this could be undesirable. An additional question is then why queues are formed and how the causes could be modelled in a simulation program or by a mathematical formula. The essential reason for queues to appear is the random behaviour of customers. The randomness of some of the processes that are involved when queues appear can of course be questioned, but from a viewpoint outside the system things may appear to behave randomly. A customer who comes to the bank with some bills to be paid does of course not think of the service he requires as random, but if all customers, and their different requests for service, are treated as samples from a stochastic distribution, we often find a very good correspondence between "reality" and samples from the distribution. Since customers arrive irregularly and have different requirements, and the server works as fast as it can, some customers find that they have to wait and a queue is formed. If the queues, and the time spent in them waiting, should be reduced, it is necessary to know something about the properties that have impact on the size of the queue and the waitingtimes. There are many system properties that determine the delays and the queue lengths. These may be more or less easy to implement in a model. Analytic models and simulation models have different capabilities when these matters are concerned. It is, formally, possible to picture any mathematical or verbal description in a simulation program. In reality, however, things are much harder. Let us leave this discussion for the moment and concentrate on the essential differences between analytical models and simulation.

As stated already in 2.2, analytic models depend on two things: the possibilities to express a behaviour as a mathematical expression and the possibilities to solve the resulting equations. If we want the result of the calculations to be a closed form solution, i.e. a solution that is not expressed as for example sums or integrals, we have to rely on the ingenuity of the analyst and his ability to manipulate the expressions obtained from the model. The possibilities to set up expressions that describe a queueing structure are usually greater than the possibilities to

solve the resulting equations. In an unsolvable case it is often possible to obtain an approximate or numeric solution. Analytic models rely on assumptions that are expressible as mathematical expressions and this can in some cases be limiting. Verbally described properties of the system, as dependences, priority and so forth, are sometimes extremely hard to incorporate in an analytic model.

Simulation on the other hand, does usually lend itself well to model verbally described properties as well as mathematical expressions. It has other drawbacks however, and these are usually merits of the analytic approach:

- a) A simulation model is complicated to describe. There exists no simple notation, as the mathematical formalism, to describe a simulation model. The program itself is therefore the only documentation available that completely specifies the model. But in the case of large models, it is almost impossible to read the program and find out how the model is built. To develop an useful model language is a coming and important research subject.
- b) Simulation models are time-consuming to develop. The ratio of modelling time to implementation and debugging time for an average simulation project is at least one to ten. In some cases there is a substantial validation time. It is not uncommon that someone who has not developed the program is the one to use it. In such a case considerable time is spent in a learning process that in the most favourable cases ends with acceptance of the model and the results it produces.
- c) Simulation programs are time-consuming to run. Simulations that employ stochastic distributions, which most simulations do, have to be run for considerable time to produce statistically significant results.
- d) An analytically derived formula can in many respects provide more insight than simulation results. It can for example be possible to see that the performance measure increases exponentially and from this that the increase cannot be compensated by for instance a linear increase of capacity. Such an observation will be hard to make if only simulation results can be provided. The analytical result may not be especially good quantitatively, but gives the direction, which at times can be more advantageous.
- e) General results, as for example for $M/G/1$, can be used in an analytic approach. This makes the model simpler to change, and can sometimes also mean that standard packages for calculations can be used.

3.2 Analysis of queueing structures

The methods available for analysis of queueing structures with respect to their capacity and performance are mainly analytical methods as queueing theory, queueing network theory, and numerical methods and

simulation. Combinations of analytical methods and simulation are also possible to use in that some parts of a simulation are analytical models. When the delay caused by a component is to be estimated, an analytical expression is used instead of event scheduling. This kind of simulation is usually referred to as hybrid simulation and has received increasing interest in recent years (SCHW 79). The other end point of this combination could be called hybrid analysis, which would mean a model in which the major part was analytic and in which a minor part, that was hard to model analytically, consisted of a simulation model. The simulation results could be expressed as for example a function, deduced from a limited set of results, perhaps by interpolation.

Analytic models and simulation can be used in combination in yet another respect. The drawbacks of the simulation technique that were brought up in the previous section, make it unsuitable for investigation of systems for large sets of input parameters. It is for example not a good instrument for dimensioning, in which case a model usually has to be run for large sets of different input data. Simulation is then used to validate a simpler analytical model which in its turn is used to estimate how a system should be dimensioned. This combination of simulation models and analytic models has also other advantages. When a particular result is obtained in a simulation, it may be difficult to find the individual component or quality of the model that is the main cause of that result. If it for example is a feedback structure in a queueing model that contributes most to the large variances observed from output, this has to be found in a simulation by omitting properties that could be assumed to have impact on the variances. If an analytic formula, although simple, that expresses the variance could be found, it could give information about the impact of the feedback. The analytical formula may however not give especially accurate values for the magnitude of the delays. Simulation is usually supreme in such a case.

Another concept that has appeared recently is hierarchical analysis (KUMA 80). In this, simulation and analytical models are used to optimise some performance of a system. Since simulation is a fairly expensive tool to use and since optimisation usually means investigation of a large number of points in some state space, optimisation with simulation is in some cases not feasible. An analytical model is therefore used to find a point near the optimum and after that simulation is used to find the "real" optimum. The number of points that have to be investigated with simulation is thereby decreased considerably. It is in these investigations assumed, without any further arguments, that the simulation model is more accurate than the analytical formula, and that the optimum obtained from the simulation lies closer to the "real" optimum than the analytical optimum.

This assumption is probably correct for the major part of all investigations, but it is not self-evident. The main argument for the accuracy of the simulation is usually that it is detailed. But a detailed simulation is not necessarily accurate. It may be that it is burdened with unnecessary details while the one pertinent quality of the system is omitted in the model. A critical investigation of the parts of a simulation should therefore be undertaken before any runs are made. This could perhaps decrease the size of the program and reduce running costs. How such an in-

vestigation could be performed for queueing structures will be treated in chapters six and seven.

There exists a variety of concepts that are used to describe the behaviour of queueing structures. Some of the concepts are listed below and then treated in turn with respect to the ease with which they are implemented in an analytical model and in a simulation model.

Dependent service times

Service time distributions

Polling systems

Forking and joining

Scheduling

Priorities

Feedback

Routing

The list is of course not exhaustive, but shows some of the major things that are imposed either by system constraints or by the system designer to improve performance.

Sequences of dependent service times and calculation of performance measures that involve dependent stochastic variables have received fairly little attention in queueing theory, mainly because of the immense problems connected with a theoretical analysis of dependent structures. It is usually easier to analyse a randomly behaving system than a deterministic, and dependences tend to lessen the randomness. The few investigations performed show that service systems with dependences have some nasty properties and that they therefore should be treated with extra care (VRAN 82).

Dependent service times are easily created in simulations. Other dependencies, such as varying service rates and state dependent routing and scheduling, do not usually create any problems.

The service time distribution plays an important part in most theoretical investigations of queueing structures. Equally important is perhaps the arrival process. The simplest analyses make use of some Markovian properties and the most common example is the simple queue with Markovian input and Markovian service times; usually with one server. Much effort has been used to extend the results for a single or many server queue to other arrival and service time distributions than Markovian. This has succeeded for some single server systems, but as for networks of queues not much progress has been made. The additional difficulties introduced by the lack of a Markov property are of a magnitude that makes most systems unsolvable. Various simplifications have been made, however, such as approximate methods and numerical solutions. But most queueing structures consist of several queues, which interconnected form a network, and the interesting behaviour seems to stem rather from the multitude of queues than from the service time distributions. Many investigations have shown that the assumption of exponential service times in queueing networks can be justified in a large

number of situations. It is therefore likely that the efforts to extend queueing network theory should be directed more towards the incorporation of some of the other items in the list above and less towards more elaborated service time distributions. Some of the results from the case study that follows do also talk in favour of such a conclusion.

To incorporate different service time distributions in simulations gives seldom rise to any difficulties. Most distributions, both discrete and continuous, can by some method be generated from a standard random number uniformly distributed between 0 and 1. Most continuous distributions have to have some mathematical form to be possible to generate and it can sometimes be hard to fit measured data to some known distribution. But measurements are often presented class-ordered and can therefore be illustrated by a discrete distribution for which there are efficient algorithms (WALK 77). Use of sample distributions can sometimes complicate the verification and validation, since analytical models cannot be used. But in some cases distributions can be exchanged to produce results that could be compared with data from analytical models.

Polling systems are becoming increasingly common as systems become more distributed. The extensive use of microcomputers has increased the need for communication, and since the algorithms for completely distributed systems have not yet reached a level, and perhaps never will, where they can compete with hierarchical, we end up with systems where some central component polls the distributed processors.

The possibilities to analyse polling systems theoretically depend on the assumptions. Some results have been published (REIS 82, KONH 74), mainly for polling cycles, queue lengths and waitingtimes for different disciplines, but much is still left to do. One problem with polling systems is that they tend to ruin some of the "good" properties of the arrival process which make them analytically tractable. The output process from a polling system is not Poisson even if the input is Poisson and if it is used as input to some other queue, a vital assumption cannot be used. Some systems do however not seem to be especially sensitive to the lack of Poisson character of the arrival process.

Simulation of polling systems is cumbersome. Polling is not complicated to implement, but time-consuming to run. A polling system scans a number of points in some specific order, and depending on the result of the poll, i.e. whether the polled system has anything to deliver or not, time to scan that point and move to the next is changing. This means that the polling process has to be implemented as a regularly evoked event and since the frequency of the process is usually quite high, a large number of events is generated. Distributed systems with polling in several layers will therefore be extremely time-consuming to simulate. And there is, since the arrival process is stochastic, no way to simplify the process.

Forking and joining is a concept of queueing structures. It appears where for example tasks in a computer or communication system are allowed to fork and to be processed in parallel. Some such models have been solved analytically (TOWS 75). Petri Nets are well suited to describe the precedence graphs, but do not imply any solutions (AGER 79). This is a new field and shows many interesting research objectives. In the case study,

the AXE system, forking is abundant and has great impact on the system performance.

Simulation of forking and joining gives rise to a number of problems. Many simulations are built as a flow of some entity through a system. If forking is present it is usually these entities that fork and form new entities that have kinship with the old ones. Entities do often have attributes that are important to the treatment and these have to be brought over to the new ones. Such parameter transmission may cause problems in terms of measurement, memory demands and process termination. Forking and joining means that all customers that arrive to a server cannot be treated in the same way. It may for example be that when a customer arrives at a server for the third time it shall split and form two or more new customers. The server has therefore to remember, or at least check, whether the customer is of the specific kind.

A simulation is much complicated by customer fates that cannot be expressed as properties of the server it visits. Customer traces have to be stored, and read at each decision about the future fate of the customer. Unless this information is coded in some way — in which case it becomes difficult to read — it will either occupy large sections of program code or demand large amounts of input data. Here as in many other cases readability and efficiency are conflicting demands. Forking does also complicate verification by use of traces since these often become irregular and interleaved. A general conclusion that can be made however is that forking is most important and usually has to be modelled.

Scheduling is one of the classic problems of queueing theory. At a first glance only FCFS (First Come First Served) may seem sensible, but since computers do not have the same objection against "unfair" scheduling algorithms as humans, a number of other disciplines have come to be used. A large number of theoretical works have been published about scheduling and references could be found in any monograph (CONW 67). Ingenious designers do now and then come up with some new and previously not treated scheduling algorithm and the analysis of the performance of a system that inhabits it may turn out to be difficult.

Simulation of different scheduling algorithms is usually straightforward. It may involve searching of queues, but most simulation languages contain elements that simplify queue manipulation. A problem that sometimes arises is preemption, i.e. when a customer who receives service is interrupted and some other given service instead. In Markovian systems this can be done easily, since an interrupted customer can be treated as a new, but for most systems preemption means that the originally assigned service time has to be remembered and manipulated in case of an interrupt. From a programming point of view this creates messy program structure with unnecessary tests and communication over global variables. More sophisticated simulation concepts and a related language could eliminate many of these problems and such have been proposed but are still at the tentative stage (STAW 82, BIRT 79).

To give some customers priority is perhaps the most common way to favour customers with urgent requests for service. Priority structures can be implemented in a number of modes, both in hardware and software. They may be preemptive or not preemptive. Modern communication and computer systems make extensive use of priorities.

Priority structures have been studied analytically (CONW 67, WALL 79). Results concern mainly waitingtimes, queue lengths and busy periods, and do in most cases make use of some Markovian assumption.

Simulation of priority structures gives rise to the same kind of problems as simulation of different scheduling disciplines. Priority with preemption is complicated to implement in a simulation and involves in most cases skip-chains and tests. Simulations with priority are time-consuming to run compared with single queues.

Feedback systems seem to have some extraordinary properties. They appear both as a service discipline (Round-Robin) (KLEI 75), and as a natural consequence of some system designs. An early theoretical work concerning waitingtimes was done by Takaçs (TAKA 63) and since then feedback systems have been treated with more or less success (DAVI 76, DISN 76). A general conclusion from these works is that feedback seems to make things "worse" in some respects and "better" in others. It can for example be used to improve the service of some customers with respect to their waiting times, especially for customers with small requests for service. But it does on the other hand increase variances and length of busy periods, which in most cases are unwanted effects.

Simulation of feedback can be done without any problems appearing on the surface. Investigations that have been performed in conjunction with the analysis of the AXE system show, however, that many interesting observations can be made for feedback systems. One of the most alarming is perhaps that the variability of results obtained for feedback systems tend to increase. This is true for variances of stochastic variables as well as variances of means. The results that will be presented for the AXE system will show that the increase, compared with systems that do not contain any feedback structures is substantial in some cases. Investigations of simpler systems have also shown that the sensitivity to deficiencies of random number sequences is dramatically increased if feedback is present. The simplest feedback system does not have any delays in the feedback loop, but it seems as if the presence of such in some cases dramatically changes the properties of the system. A system with delays in the feedback loop is hard to analyse analytically however, and so far few results have been published.

The results due to Takaçs (TAKA 63) assume a probabilistic feedback decision. It may be argued that such a model does not picture the behaviour of real systems since Takaçs' model could mean that customers make an indefinite number of turns which hardly is a realistic model. New results, independently obtained by Eklundh and Rapp (See appendix) and Lam (LAM 81), show that in the Markov case delay is independent of the scheduling decision.

Feedback does therefore seem to be one of the most important properties of queueing structures and has to be modelled in detail. Simulations have to picture the individual sequences that are traversed in a feedback loop and carefully estimate the delays encountered.

Routing is in analytical models assumed to be mostly probabilistic. The possibilities to model deterministic or alternative routing are very limited. Queueing network models can however model routing to a certain extent by class switching, but the applicable service time distributions are limited. Dependences and forking are also hard to model.

Routing in simulations does not represent any problems. It can in some cases involve rather large routing matrices, but these can usually be handled.

The analytical models described above do all apply the same kind of system view. To be able to analyse a system, a model is built with the aid of certain mathematical tools. An essential descriptive tool is the distribution used to specify for example the service time or the interarrival time. Lately some objections against this system view have come up however. It is argued that it is generally impossible to verify the existence of a specific distribution and that only quantities that can be measured should be used in an analysis. This "new" approach is usually called Operational Analysis and was first proposed by Denning and Buzen (DENN 78). It makes use of a number of measureable quantities and derives a number of operational laws for these. Regular queueing theory measures like mean queue length, mean waiting time and their relation via Little's result are possible to obtain but have a somewhat different appearance. It is not generally agreed that Operational Analysis really is a "new" approach, but it pinpoints some problems that indeed have to be addressed in any study of queueing structures and are relevant to simulation as well. Operational Analysis will be used in chapter six to make a preliminary investigation of the relative importance of different system components and a more detailed description of the technique is therefore postponed till then.

3.3 Obtainable results

It would of course, already when simulation is considered as an instrument to investigate a system, be satisfying to know whether the properties that are of interest could be obtained by simulation. Since simulation is a fairly expensive method to analyse a system, much could be saved if unnecessary investigations need not to be embarked on. In a general case it is hard to give any list of results that could be obtained, but for the limited case of queueing structures some hints could be given.

There are two main objectives of simulation to be distinguished: estimation of performance measures and investigation of structural properties. The former is further divided into two groups: quantitative and qualitative measures.

To investigate the structural behaviour of a system means to look at how the system performance changes if the structure is changed. The difference between structural changes and other changes may sometimes be subtle but it is in most cases easy to tell which view that applies. Take for example a multi processor computer system. If in one case all processors are allowed to share memory with all other processors and in another each processor has its own memory, we have two structurally different cases. But if the size of the programs that run on the different processors is picked from different distributions at different times this will mean no structural difference. It can informally be said about structural changes that they occur in leaps only and not continuously. In the

former case we are interested in comparing the case with shared memory with the one without. But to do this we have to have some standard of measure. If we say that one system is better than another we also have to say in what respect it is better. It could of course be that one of the systems is better than the other in one respect while the other is better in some other. The interesting thing is however, that we are not so much concerned with the absolute value of the measure but more with the relative magnitude compared with the other structural case.

The two cases usually differ in how detailed the model must be. A structural investigation can in most cases be performed with a less detailed program. But the same measures have to be used to compare different structures as those used to estimate other results of the simulation. It therefore becomes interesting to analyse which these measures, in the case of queueing structures, can be.

The basic performance measures for queueing structures are waiting times and queue lengths, given either as moments or as distributions. Apart from these utilisation and busy periods may be of interest. It is of course also possible to have interest in special properties of the studied system, but these properties do usually not refer to the queueing structure as such. The most common measures are means and variances, given within some confidence limits. The literature on statistical analysis of simulation output is abundant and deals with some different approaches, of which the autoregressive and subrun approaches (FISH 78), the regenerative approach (IGLE 78) and the batch means approach (SCHR 79) can be mentioned. If the model is fair, the above methods provide us with good instruments to estimate the significance of our results. Simulation does in this respect usually not cause any problems. It is of course possible to perform the measurements incorrectly, but that has to be ruled out in the present discussion since no general method can be found to prevent such errors. But it is not always means and variances that are the only properties of interest in a simulation. What are the possibilities to measure other quantities as well?

The ease with which measurement can be performed decreases with load however. This has not been proven generally, but the results due to Olsson (OLSO 76) show that for a large class of systems measurement are quite troublesome.

Traces, i.e. printouts from simulation runs that show how the state of the simulation changes with time, are important to the validation process as will be seen in chapter four. But these traces are sometimes brought up to show how "reality" behaves. A realisation, showing for example the history of a customer in some server system, is sometimes used to verify the possibility of such a customer history "in reality". Could such implications really be made?

The problem falls back on the possibilities to measure rarely occurring events. An event in a simulation is when the model changes state from one state to another. Depending on how we define the states, events that we are interested in will occur more or less often. Sequences of events are more unusual than single events and tails of distributions can be hard to measure with good statistical significance. It is good practice to in advance consider how likely events are and how long the simulation has to be run for a specific number of events to occur.

Another question that arises in this context is whether simulations could be used to discover events or series of events that cannot be discovered in any other way. Is it for example possible to build a model of some human behaviour and to run the resulting simulation program in order to find out what people can come across. Generally, the answer has to be no. Primarily because the simulation investigation becomes in much a random process, since the result is not anticipated in advance. This may seem a peculiar statement since if we could say in advance what the results should be, we would not need simulation at all. But there is quite some difference between the anticipation of an average queue length and the anticipation of a specific fate of a person or some other entity in a simulation. The existence of an average queue length can be anticipated, but it is hard to estimate its magnitude without some additional instrument like simulation. But if a specific fate of a human being is anticipated in a model, it should be possible to deduce the possibility for such a fate to exist without using simulation. It is of course possible to write a simulation model and run it for some different input data and to see what comes out of it, but the question is whether it would not be easier to get the insight the simulation can give by just examining the situation. It is at any rate a rather irregular way to perform an investigation.

But if simulation, although it is not advisable, is used as described above, it puts some extra load on the verification and validation processes. Since the pure existence of an event is the prime result of the simulation it has to be carefully checked that this event is not the result of a programming error or faulty model. In such a case the result would be quite useless and it is not easy to deduce the correctness from similar cases since they all are unique events. A fate of say ten sequential events picked from ten equally likely states gives a possible state space of ten raised to ten sequences. To produce them all in simulation runs would require considerable computer time, in spite of the example being fairly simple what the number of states is concerned. There are other methods, like for example enumeration techniques, that could be used to list all possible events. Simulation does not seem to be the technique that should be applied in cases like these.

But even if all the objections in the above paragraph are rejected, problems still remain. If the existence of an event is of vital importance to the accuracy of the simulation model, there must also be strict demands on the accuracy of the model and the modeller. Is for example input data and model sufficiently accurate to say anything at all about the occurrence of a similar event in "reality"? The interesting question is of course how likely an event is or how often it happens. But if a statistical significance should be added to the number of runs required to create the events things go astray completely. The conclusion is therefore that investigations of this kind are hard to conduct with simulations.

In order to be able to make any statements based on the results of a simulation, we have to convince ourselves that the model indeed is correct. We gain this confidence by a series of investigations, and the individual steps are treated below.

4. Verification, validation and credibility.

4.1 General

We are all inclined to make errors now and then. We discover some, but many do never come to our attention. But some errors can of course be fatal, and if we fear the effects, we try to minimise their occurrence. This can be done in many ways depending on the kind of activity we are involved in, but one common way is to try to check the effects of some invention or construction before it is put into use. Someone who has built a lift tests it first without passengers and does perhaps overload it to see if it can stand the tensions. Motorcars are tested to reveal their ability to resist forces appearing in collisions and to find out whether someone who was in the car at such an occasion would be damaged.

To-day more and more functions in our society rely on computers and computer controlled activities. Information is gathered and fed into computers and decisions are based on the compressed and extracted results obtained from processing of this information. Money transactions between computers are getting more and more common and less money is exchanged directly between people to-day than before. Computers aid people on all levels and the engineer and the secretary are both depending on computers.

An important question in this context is whether the advice and guidance we get from the computers are correct. We know that errors can occur: people are asked to pay taxes on income they have never had, and some are said not to exist because they cannot be found in the data bases.

The computers we use have to be taught to follow a program, and since the advent of computers much have happened in computer pedagogics. The languages with which we communicate with the computers have developed and are to-day easier to understand than they were before.

Concepts at high levels can be brought into a language, and we have witnessed the rise of many different dedicated computer languages over the last two decades. Some of these have been addressed to simulation and a few were mentioned in 2.3.2. These languages have enhanced our ability to write programs that picture our intentions in a correct way, but this is not always sufficient in a simulation context, since the computer always does what we ask it to do, but that is not always what we intended.

For a simulation model to be a good description of the studied system, a number of requirements have to be met. We have to transform our perception of the system to a model, write a program that depicts the model and then run the program to find out something we did not know before. This may seem simple enough, but each step is error prone and we are by no means good at the transformations. Everything we do has to be checked and rechecked and it is not seldom we find fatal errors. The fi-

The above two forms of program and model checks (verification and validation) are obvious, but there exists in addition a third step that deals with whether the model can predict anything for configurations and situations that are not validated. This is usually referred to as the credibility of the model.

This chapter is devoted to a discussion of the problems encountered in the Verification, Validation and Credibility processes.

4.2 Verification of Simulation Programs

Verification of simulation programs does not in general differ from verification of other programs. It means in essence to prove that a program is correct and in correspondence with the requirements. But simulation programs have some extra difficulties.

Simulation programs employ stochastic distributions to realise the intended behaviour. This means that an execution not always will give a "typical" behaviour, but sometimes a rather odd or unexpected realisation. It is therefore essential to be able to put the program in a "corner" in which the realisation can be determined in advance. This requires the program to be constructed in accordance with some basic principles, of which some apply to all programs and some are specific to simulation programs.

Structural programming is a concept that appeared in the early sixties (DAHL 72). It later came to mean approximately to program with the use of procedures and has to-day that meaning. To accomplish a structured program, support has to be given for such constructions. It is possible to write structured programs in FORTRAN, but it is unfortunately also possible to write extremely messy programs in this language. The most crucial part from a simulation point of view is the lack of data structures.

There are some basic demands that have to be put on a good simulation language, but once these requirements are met, the choice of language is not crucial.

A versatile simulation language has to be well equipped with concepts that support organisation of data structures. It seems as if this is one of the forgotten issues of simulation modelling. Well structured data is sometimes the clue to simple and handleable programs.

A second important feature of a simulation language is dynamic variables. These are a must for handling components of the simulation, which are created, last some time and eventually die.

The language should finally be supported by an advanced compiler and runtime system, which can aid the modeller/programmer during the programming phase.

A step in simulation language construction is to develop primitives that can be used in a large class of simulation problems. It seems as if the hig-

best reasonable level at the moment is to furnish the language with standard routines for scheduling and queue handling. These routines should not be too context dependent, which is the case with some of SIMULA's concepts.

Attempts to automate for example data collection and statistical analysis should not be extended too far since these undertakings are problem related. With a good programming methodology, measurements do not constitute any severe problem.

Programs that are used to calculate some quantity can usually be checked against some special case for which the result is known. If a large number of algebraic manipulations in the program result in a correct value for some input it is likely that the same will be true for some other input if the manipulations are the same. This is the way in which most programs are verified. The attempts to prove program correctness more formally have not been successful.

Simulation programs can usually not be verified in the same way. Depending on the random number generators, different executions can result in different values of a measured quantity, and it may not be possible to judge whether the value is correct or not. To remove the randomness from the distributions used in the program is not especially fruitful, since in such a case the execution for a given input will have only one outcome which perhaps is not at all interesting. Take for example a queue and a server that act as a stable D/D/1 system. Such a system has a totally predictable behaviour and many interesting situations do never arise: There is no queue, a customer cannot arrive when the server is busy, departure and arrival cannot occur at the same time, and so forth. It is in most cases situations like these that are erroneously programmed. It is therefore essential to try to force the program to execute as many combinations of events as possible.

The first step of the verification process is to remove the programming errors that are discovered by running the program. The second step is to verify that the model is indeed mirrored by the program. The following example shows the difference between these two verifications.

To discuss these two steps, let us consider a priority queueing system. The priority structure is assumed to be preemptive-resume, i.e. a customer at a lower level is immediately interrupted by an arrival at a higher level and the system continues to serve customers at the higher level as long as any are present. A program, which should illustrate such a system, could be incorrectly programmed in a number of ways. The program could for example assume that whenever a higher level becomes empty, customers are present at a lower level. Such an error would eventually be discovered since even if the system is run at a high load it will at times become empty. A short test run could however very well succeed due to chance.

Another error could be made that resulted in incorrect updating of the interrupted customer. When a customer is interrupted it has usually received some service already and has consequently less left than when it first got service. When the interrupt occurs, remaining service time should be updated and service resumed later with this new service time. If this is forgotten and the customer is left with its original service time

nothing in particular will happen to the program unless the mean service time for the customers at the level at which the error is made has extremely high value. In such a case measurements may reveal the error, but otherwise it may be extremely hard to discover.

The two errors related above differ in principle. The first will be discovered if the program is run long enough and under sufficiently diverse conditions, but the second can never be discovered in this way. The necessary verification is consequently different in the two cases. The first is discovered by regular program testing methods and by tests that do not involve measurements, i.e. sampling of states and subsequent statistic analysis. This first type of verification is here referred to as type I. It contains verifications such as traces, counting events, comparing the number of generated customers with the number that has left the system, and similar methods to follow the behaviour of the simulation.

The second error can only be discovered by mere chance or by comparisons between known, for example analytic, results and simulation results. If there exist analytic results, and the simulation model, after several runs, refuses to yield results that can be considered statistically significant and agree with its counterpart, errors have to be suspected. But not all models can be verified with analytic results, and in such a case there exists no known method to verify the model. The example above for instance will not reveal any error if exponential distributions are used, since the resampling will have no effect in that case. And even if there indeed is a difference, it can be hard to discover. Mean values of measured quantities may be rather insensitive to errors of the second kind above. Higher moments do often have to be compared to bring forth the mismatch, but these are more cumbersome to measure and the standard techniques available to estimate the confidence limits for higher moments are not simple. Small test runs do not usually produce limits narrow enough for verification purposes. Extreme care has thus to be taken to verify complex queueing structures.

Some errors, which cannot be discovered during the initial program tests mentioned above, can be found if the results from the simulation are checked against each other with the aid of Little's result. If the arrival rate to a subsystem of the simulation is known (or can be measured), and if delays and queue lengths are measured, then the means of these quantities have to relate in accordance with Little's result. This check is extremely useful for discovering wrongly implemented measurements, since it is unlikely that both delays and queue lengths are incorrectly measured at the same time and if so that the errors work in the same direction. Little's result is very general and can therefore be used for almost all kinds of systems independent of the distributions employed (KLEI 76). This type of verification does not however say anything about the correctness of the components in Little's result.

Verifications that involve some kind of measurement and comparison of obtained results are subsequently called type II.

4.3 Validation of simulation models

Even if the verification process is considered successful, and the modeller is convinced that the program and the model agree, further tests have to be made. Nothing has been proven by the verification process about how well the model agrees with "reality". Before any confidence can be put in the model, results from this have to be compared with results obtained from measurements performed on the "real" system; if it exists. This process, to calibrate the model against "reality", is called validation.

Validation involves measurements, both in the simulation and in "reality", and is consequently subject to errors occurring in these.

Interesting questions are which performance measures to compare and how extensively the simulation model should be validated. Something has of course to be left to the simulation since there is not much point in building a simulation that in every respect can be confirmed by measurements. It would in such a case be better to use the measured data directly than to rely on the simulation.

The use of simulation models of queueing structures has some definite goal. Some property, that would be hard or impossible to investigate directly, is of interest. Such a property is often the capacity of the system, i.e. the system's ability to carry out the tasks it is assigned or may come across.

Computers are nowadays often used to control processes. It may be an airline booking system or some chemical process. In spite of the decreased cost of electronics and computer devices, large computer systems are still expensive and on a competitive market someone is always prepared to make a cheaper version of some product if other vendors have wasted hardware or software. Systems are therefore made with as little excess capacity as possible. Another reason for insufficient capacity is that load increases over the operational period of the equipment. Systems are therefore, some time during their operational period, run on the brink of their capacity, but are usually considered inappropriate if they exceed this limit.

It therefore becomes important to know at which point a specific system reaches its ultimate capacity or, if this is specified, whether a design can meet the demands put on it. Waiting times at the airport booking desk have to be short and the chemical process has to be properly supervised.

If simulation shall be used to find answers to questions like these, the system has to be investigated in heavy load situations. That is to say, the simulation has to behave as the system not only in normal situations, but also in situations where properties, that are of little or no importance when the system is lightly or moderately loaded, may be of vital importance. This means, that the simulation has to have what is usually called structural validity, a concept that will be discussed further later on. It is, at any rate, not sufficient to validate the model for situations in which almost any model with fairly good system descriptions would be sufficient, but the model has to be validated for heavy load situations where new and unexpected properties may present themselves.

The applicability of simulation is vast and the validation problem does consequently become immensely diverse and comprises a great number of different questions and issues depending on the particular application. Sargent (SARG 79) discusses a number of application areas: is the model to be used in research, development or design; what kind of behaviour is to be investigated, transient or steady state; what is expected from the model; and a quite extensive list of other subjects and applications. The questions put are pertinent, but many are based on a belief that simulation can be used for all these purposes and that the validation thus is possible and necessary. This may be questioned, as is done here, and it is at any rate not possible to find a general validation technique that can be applied to all simulations. This is at least not the case at the moment, and general discussions on validation do therefore become just too general and of little value in a concrete modelling and validation situation.

Zeigler (ZEIG 76) states three levels of model validity: replicative validity, predictive validity and structural validity. These levels are ordered in increasing accuracy, so that a model which is replicatively valid is cruder than a model which is structurally valid. A replicatively valid model can, within a defined accuracy, reproduce system data; a predictively valid model can predict future data and the structurally valid model mirrors the real system's internal behaviour. As for queueing structures, structural validity is in most cases necessary, since it is to overdo it to write a simulation just to reproduce some already known system data. And moreover, such data could in most cases be produced by an analytic model, which is easier to handle and cheaper to run.

Data available for validation of queueing structures is sometimes, and perhaps usually, scarce. Measurements are in simulation programs performed at points where the target system is not easily accessed. Measurements must perhaps be carried out in hardware and the interfaces needed are expensive and complicated to design and build. Measurements can sometimes be performed in software, but there is a great risk for the measurement to have impact on the system and thus to give biased results. This is a sort of uncertainty relation as the one in quantum mechanics, where the result of the measurement indeed is the measurement itself and not a system property.

Zeigler does, as already mentioned, group the validation techniques into three groups, but there are other ways to categorise validation. Sargent lists a number of techniques, but some of these seem to belong more to verification than to validation. Zeigler's grouping is based on the result of the validation, while Sargent's deals more with the way in which the validation is performed. The question is of course whether program checks should be regarded as verification or validation, since the verification process indeed gives the modeller a certain degree of confidence in the model. Verification is not totally detached from the context and the modeller does, consciously or not, compare the program with the original model. This becomes more clear if we look at the figure of model development. Verification was indicated by an arrow from the program box to the model box, and validation in the first meaning as an arrow from the "result" box in the right branch to a similar box in the left branch. This view is a bit too simplified. The figure below is an extended version of the already shown and can perhaps shed some extra light on the discussion above.

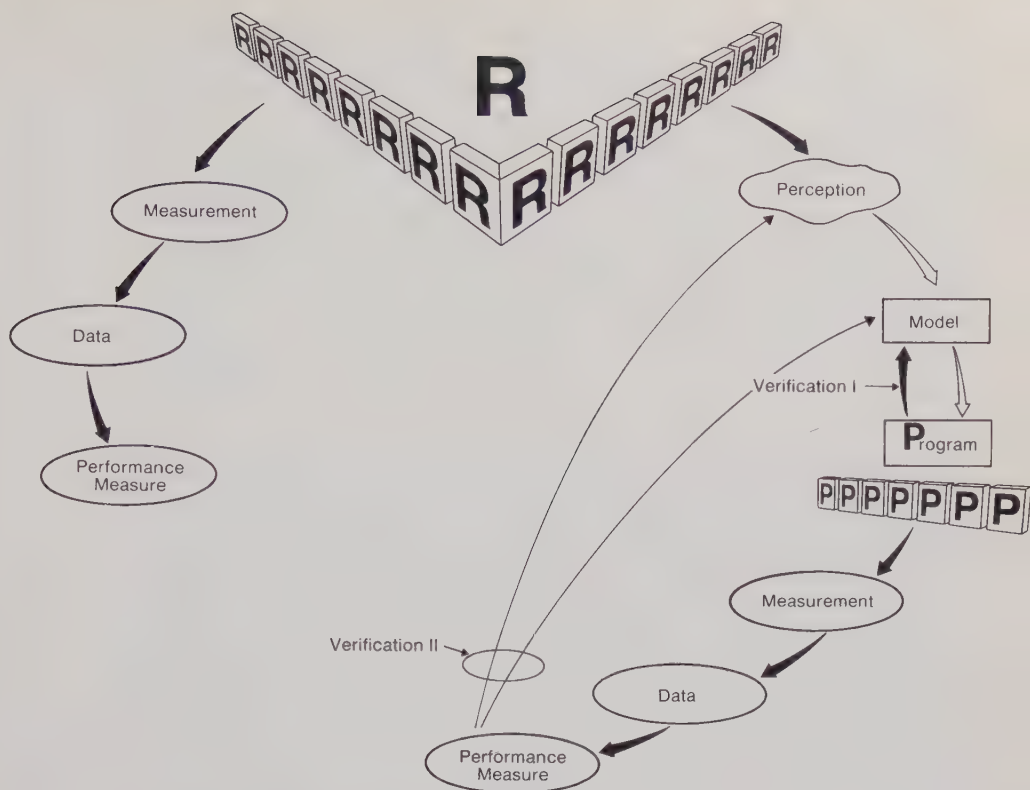


Fig. 4.2 Verification types I and II

The terminology is to some extent unclear, since intuition, measurement, formal and informal methods are mixed and used interchangeably. Almost every author who deals with the subject introduces, probably based on the special applications he is involved in, a special notation that then is assumed to be valid for all kinds of simulations. This has so far not been true, and there is no point in adding another few concepts to the already mixed-up pattern. The terminology used here is biased by its application, queueing structures, and any new notation is ad hoc only.

Some validation deals with the model directly and some with the results obtained from it. Face validity, traces and event validity, as defined by Sargent, can be said to belong to the first, while internal validity, parameter validity and comparisons to other models belong to the latter. Face validity does in fact belong to both groups, since it can be either the model directly or the results that are compared with our knowledge about the system.

The most efficient validation comprises measurement on the "real" system, but this kind is also the one most sensitive to errors both in the simulation and in the measurement process. Since direct comparisons involve all the steps in measurement and modelling they are subject to the same kind of errors as discussed in 2.4.

The discussion so far has assumed the existence of a system which could be used as the "real" world and with which comparisons could be made. This is not always the case, and simulation does perhaps have its greatest advantages when used in a pre-design phase to check alterna-

tive designs before any real system is built. In such a case no validation object exists and the arguments above do not apply directly. Combinations of simulation and hardware are becoming increasingly common (BEAU 79), and "hardware in the loop" simulations have been used for a long time in continuous simulation. The simulation is in these cases used to illustrate some yet not existent part of the system, and can not be a part of the validation either. Validation of conceived systems has to be performed entirely within the right branch of the figure above and any formal methods do not exist. But the situation is on the other hand more favourable, since it usually is not the exact value of an interesting quantity that is of importance, but rather its relation to a corresponding measured in some alternative system. The ambiguity of the assumptions may not permit any accurate predictions and the model can consequently not be used to predict the final system behaviour.

Another question is whether all programs and models have to be verified and validated. It can be argued that a model should be used only to investigate how an idea works or how some unrealistic social model turns out. And in such a case there is nothing to validate and verify. The first may be true, but the latter, i.e. that there is no need to verify the program, must be wrong. If someone builds a simulation model, it is to be used for some purpose, and some kind of information is to be extracted from it. If we do not want the information to result from programming errors, the program has to be verified.

4.4 Credibility of simulation models

Validation is more difficult than verification and credibility is definitely more difficult than validity. To say that a model is credible is to say that the model is able to express something we do not already know and cannot, with reasonable effort, find out without the model. We enter in cases like these a world different from the one we have moved around in so far. We move from deduction to induction, from "be sure" to "believe". Validation is always to some extent based on intuition and more or less convincing arguments, and when it comes to credibility, nothing else is at hand.

Most of the opinions we have about life are based on induction. Induction is a strong argument if based on a sufficient number of observations, and it seems as if knowledge and induction, or generalisation, are interchangeable or at least have much in common. Some things are self-evident and we find no reason to doubt them. But it is still not obvious, in a philosophical sense, that certainty exists and that some things are "true" (MOOR 59, WITT 69, GARD 81). Our minds are in a way integrating or averaging, and when things have happened a sufficient number of times we are inclined to say that so and so is life. But this averaging does of course have risks. It is usually easy to find a case for which the statement is not valid. To find someone who has experienced that the sun did not rise one day is perhaps difficult, but smokers do usually have an uncle or grandfather who smoked forty handmade turkish cigarettes

a day most of his life and who did not get lung cancer. Thereby, they believe, having made the argument that smokers in all controlled investigations have a greater tendency to get lung diseases quite worthless.

Transformed to simulation it becomes a question of when a model, and the attached program, are credible. When do we believe what the model says, and when do we have enough confidence in the results to dare to act according to them. When we enter this domain, nothing is true in the respect we are used to. One modeller says he believes in the result of the model, someone else can say the opposite and neither can prove he is right. In this context rather odd things as experience and giftedness become important and only the future can distinguish between good and bad models.

The modeller who has built a not so good model does sometimes defend himself and says that the presuppositions were not as he had thought they were. This can of course be true, but the fact is perhaps that we have to accept the difficulty of modelling and realise that now and then we build models that do not behave nearly as the system they should be a model of. Something was perhaps extraordinary with that particular system, and our technique and experience did not allow us to make any better model than we did. This has to happen all the time and there is not much we can do about it. If we therefore try to find some fairly general rules that could be followed in a modelling process we are bound to find exceptions and anomalies that do not fit into the scheme. The choice is between rules that apply in some cases or no rules at all. Those who are against rules and who argue that there is no point in having rules since they have too many exceptions are probably not aware of their own way of thinking and that they, perhaps unconsciously, have their own rules that they sometimes interpret in one way and sometimes in another. It must, in such a case, be better to have rules that one is aware of do not apply in all cases and which sometimes are obviously wrong, than to have inconsistent rules that are impossible to explain and put forth.

When a model and a program exceed certain limits even the modeller and the programmer get doubtful and are not entirely convinced that the model and the program are correct. Their experience tells them that the likelihood for errors is great in spite of all the tests made on the program and the model and they do therefore not really trust their own work. This seems to happen to all modellers and the consequence is sometimes an almost negative attitude towards modelling.

The remedy for this is probably smaller and less detailed models. Large overall models are perhaps possible to build, but do very seldom get used and do not, as intended, give anybody but the modeller increased insight into the problem. In spite of the unwillingness to report on unsuccessful projects, examples can be found (HYPH 81, EKL 81a), and once the problem is mentioned it turns out that quite a few have comments to make on the issue.

The program, the model and the result description techniques are currently not elaborated enough to ensure credible overall models of large and/or complex systems. Not even an insider is, after working with a model for perhaps years, confident that the model works and can be relied on, not to mention an outsider, who finds the barriers surrounding

the model and the program almost impenetrable and who consequently does not care to use the model, independent of how good it is.

Smaller and less detailed models could perhaps more easily overcome these problems, but would, at least at a first sight, also become less accurate. Economic models for example, for which an accuracy of some tens of a percent is required, must probably comprise many small details to be reliable. During the sixties, when the economy of the West grew steadily, models built were able to forecast the next year's growth within very small limits, but to-day, when the economy jumps up and down in a seemingly unpredictable manner, models are not able to say anything specific about the turnout of next year. This is almost like forecasting weather. If one says that the weather to-morrow will be like it is to-day, there is some sixty percent chance that one is correct. And this is what happened during the sixties when economy grew almost linearly.

This is an example of how models are built. Modellers do, without really knowing it, leave out details that they believe are not important, and which are not, as long as they remain constant. But when things change, as they have in economy over the last two decades, models which were formerly quite accurate become almost useless. A predictive model that sets the annual growth of economy to something between zero and two percent is a very accurate model, but it is almost worthless from a planners point of view since two percent of GNP is quite some money, in Sweden currently some 10 billions crowns.

If now simpler and less detailed models are the solution to some of the problems related above, how should they be built? When a modeller does not see any reason to limit the size of the model there exists no crucial selection problems, but when a model may not exceed certain limits the question of what to take in and what to leave out arises. The answers to such questions are of course coloured by the views of the modeller and all the predetermined opinions he has as discussed already in chapter two. But generally speaking, an accurate model of a system, which is not allowed to take in all aspects of the system, should quite obviously comprise those components and relations of the studied system that are of importance to the interesting property. Such a statement is a truism, and has to be specified to become of any use. It is easier to give rules for limited areas of application, but these are on the other hand less useful. The optimum is to give specific rules that apply to as large areas of applications as possible. Chapter six will deal with some relevant properties of queueing structures that may be of importance in simulation studies.

Before these issues are discussed, it is necessary to relate the development of an average simulation in order to specifically discuss modelling. The coming chapter does therefore deal with the simulation of an computerized telephone exchange.

5. A conventional approach

5.1 A specific task

This chapter deals with the simulation part of a greater project that aimed at the analysis of the control system of a computerized telephone exchange: the by Ericsson developed AXE exchange. The previous chapters have in general terms described some of the experiences gained during this and other projects, and it is obvious that the lessons learned would make another attempt to simulate the AXE develop quite differently. But much of what have been said and will be said in some of the following chapters make sense only if compared with the common way to perform simulation investigations. After the completion of a project it is usually simple to say how things should have been done and what was unnecessary work. The best thing would be however, if one in advance could state something about the possibility for a project to be successful or how much effort an investigation would probably need. This chapter is therefore devoted to a relatively detailed description of the simulation project, without any discussions about the subsequent analysis which revealed the intrinsic properties which were later to be observed as the main causes of the results. It may seem odd to describe failures or unnecessary work, but as mentioned in chapter four, there is for the moment no other way to learn anything about how to conduct a simulation experiment than to look at other attempts and to see what went wrong.

Some of the errors that occur are of course very simple and although they appear in every simulation experiment, little can be done to eliminate them. Such errors are for example programming errors, incorrect input and all kinds of misconceptions that show up during program development and the first test runs. Other lessons are too limited and apply mainly to the specific object of the investigation. If the same experiment would be repeated, these errors could be avoided, but this has no general implications. But there remain, after this sorting, some general ideas that could be conveyed through the example and these will be brought up in chapters six and seven. For this to be possible, what follows in the rest of the chapter have to be said.

Telephone exchanges are used for connecting subscribers to each other. They have developed from simple operator controlled switchboards to computer controlled switching systems, and are to-day used for a multitude of traffics, such as data, voice, telefax and others. With the rapid technical development, new services have been added and many more will probably appear in the future. The electronic revolution has made this possible by making the components, that are used to realise the new as well as the old functions, cheaper and cheaper. But an exchange is in spite of this quite expensive, and should be built to a maximum of efficiency at as low cost as possible. Low cost can in this context mean a lot of things, and such demands as serviceability, reliability and enlargeability have to be considered.

From the subscribers point of view, the exchange should provide reliable and fast connections, and the subscription should not be too expensive. The telecommunication administration likes the exchange to give their customers good service and to be as cheap as possible. These demands are to some extent rivalrous and the system has to be "optimised" to make a good compromise. The final system design is of course a combination of many different aspects that cannot be brought up here, and the simulation results do not give the final answer to any design problem. Any solution suggested by such results can be overruled by other demands that are more important, and the conclusions made here do therefore apply to the limited context of the control system only and do not say anything about the overall structure and design.

The international standardisation organisation for telecommunication systems, CCITT, has issued a number of measures to estimate the grade of service given by a telephone network. Examples are Dial Tone Delay, i.e. the time elapsed between subscriber offhook and dial tone; the time it takes to establish a connection after reception of the last digit and a number of similar measures. These measures depend on other measures that can be obtained for an exchange and can be estimated if these are known. The Dial Tone Delay for example depends on how fast the exchange observes offhook, and how fast it can assign necessary equipment to the subscriber. In a computerized exchange this assignment depends on how fast the computers in the system work and on how the work is organised. The measures above are used as arguments in contacts with potential buyers of the exchange and it is therefore essential to be able to estimate them. When an exchange works far below its ultimate capacity, it is easy to fulfill the requirements. It is when the exchange becomes heavily loaded that problems arise. Since the price of an exchange depends, partly, on the number of lines connected to it, that is roughly the number of subscribers, administrations like to buy exchanges that are sufficiently big, but not too big. This means that the exchange, at least some time during its lifetime, is to be run near its optimal capacity and it is assumed to perform well at this load.

The goal of the aforementioned project was twofold: it should analyse the control system of the AXE system with respect to its capacity and it should develop tools that could be used to dimension an exchange. There were two main lines that could be followed: analytical models and simulation.

*Apart from the measures discussed above, a number of interesting properties can be examined in a control system of this kind. The available buffer space is in all practical situations limited, and if a buffer becomes full this will in most cases result in intervention by an operator or perhaps even in a system breakdown and a restart. In such a case calls are lost and the subscriber feels that the system is unreliable. Occasions like these have to be few and the buffers have therefore to be sufficiently large, but not too large since memory space costs money and has to be controlled. There are more examples of the same kind, but they are more easily explained when the structure of the system is known and will be delayed until after a system description.

Measurements are costly to perform on a real system, and a simulation model had therefore to be used to validate some other models. Some

measured data had of course to be used to validate the simulation model, but data could more readily be obtained from a simulation than from measurements on the real system, and the simulation model could also be altered to fit the analytical models in order to investigate pertinent behaviour, such as assumptions of independence and exponential service times as will be seen later. The simulation model discussed here had therefore to be detailed enough to incorporate all aspects that would be investigated with analytical methods.

The target system, AXE, was from the beginning totally unknown to those involved in the project. A first step was therefore to collect information about the system and the processes that were implemented in it. For a system of AXE's complexity, this is by no means a simple task. The documentation consists of tens of thousands of pages and is sometimes not easy to read even for an expert. The process of information acquisition took consequently quite some time and involved a lot of selection since it was impossible to read everything that was written about the system and its functions. Much was indeed unimportant and could without hesitation be skipped, but other things were skipped and did later show to be of importance.

The project started when the system was already designed and the main goal was not to come up with ideas about the design or the layout, but more to learn something about the behaviour of systems like these and gain knowledge for future tasks of the same kind.

5.2 The AXE system

In older telephone exchanges, desired functions of the control system are usually realised with relay technique. This means that alterations have to be made in hardware by actual rewiring. Such systems are expensive, spacious and inflexible. But up till the advent of electronic computers during the sixties, there were no other ways to build control systems.

When computers entered the scene, there were many who realised their potentials for control purposes, and some attempts were made to implement computer controlled switching systems.

A switching system that uses computers for decision making differs from a conventional system in that the decisions are the result of an execution of a program in a computer. The technique to control systems with the help of programs stored in a computer is usually called Stored Program Control (SPC), and modern telephone and data switches are built according to this technique.

An exchange can thus benefit from the extremely rapid development of processing power and memory capacity of computer technology. The system is, compared with older technique, easy to alter and enlarge. Exchanges can be made smaller and less energy consuming and may be built to incorporate new services. There are a number of other advantages, like maintenance and other handling merits, that may not be ap-

parent to the subscribers, but which are of great importance to the administration.

The computer of a computer centre is, from the user's viewpoint, a general purpose machine. It can usually execute any program written in some for the computer understandable language and it delivers the results when given enough time to perform its task. The user awaits the results with more or less patience but can in most cases make use of the results when they arrive. A dedicated computer that works in a control environment has somewhat different requirements imposed on it. Certain time limits must not be exceeded and some work has to be done with priority. Take for example the situation where a computer is used to analyse digits that arrive as pulses from some subscriber's telephone. The pulses arrive indepenent of the situation in the system, and the computer must have time to receive them and analyse, or at least store, them, they are otherwise lost and the subscriber will not get his call established. To be able to accomplish such functions, a control system has to be organised with special attention to these problems. There are many ways to do this and the solution that will be presented below, in the shape of the AXE system, is only one out of many possible. The presentation focuses on the control system of AXE, an approach that can be questioned since there are many other, perhaps equally interesting and difficult parts; but since the presentation shall serve as a base for the subsequent discussion on simulation of systems like this, the approach can be defended.

The capacity of the system, i e its ability to handle the tasks it is assigned, is assumed to be limited by the capacity of the control system. There may very well be other system parameters, like for example hardware equipment besides the processing systems, that may limit the functions. It is therefore from a relatively narrow angle the system is presented and the presentation does not claim to be complete.

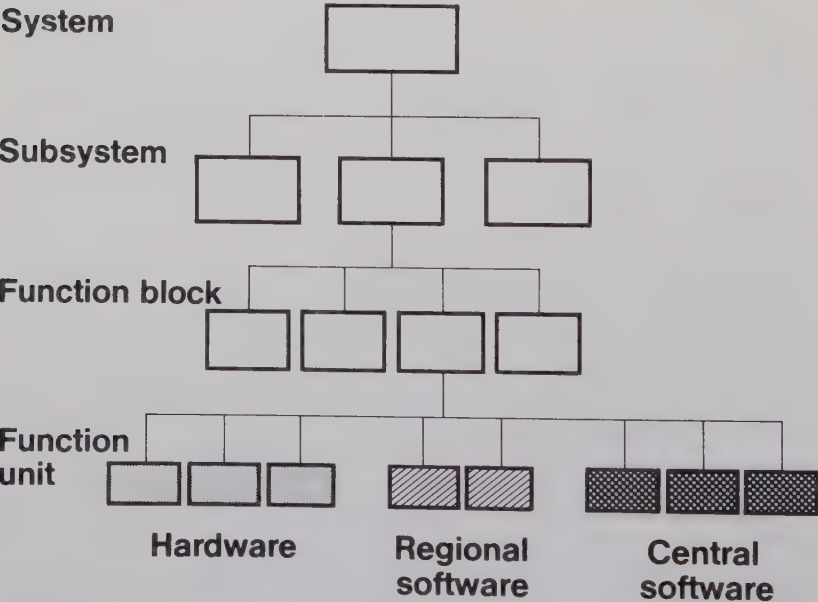


Fig. 5.1 AXE functional structure

The AXE system is an SPC system designed for switching different kinds of traffic. The presentation here assumes the exchange to work with telephone traffic, but this does not limit the presentation to any larger extent. The system is hierarchical in that it can be divided into pieces in accordance with figure 5.1 (the presentations quotes extensively from (ERIC 76)).

AXE consists at the top level of a switching system APT and a data processing system APZ. This division is based on the functions of these two parts and does not give any information about the physical structure of the system. This way to describe the system turns later out to be a complication when a simulation model is developed, but it has also advantages, such as the collection of similar functions in one module, independent of their implementation, whether in hardware or software.

APT is the switching part of AXE and consists of six subsystems, as shown in figure 5.2. Each of these subsystems is made up of software or hardware or combinations thereof. The software functions can be either regional software or central software. Regional software takes care of simple but frequent or time-consuming functions while central software performs more complex functions. Each subsystem is divided into a number of function blocks, which is the lowest level at which software and hardware are functionally mixed since a function block is made up of a number of function units, each implemented either as software, central or regional, or as hardware.

All software functions that are implemented as central software are executed in the same processor, and this central processor does therefore functionally belong to a number of function blocks. The same apply to the regional software, but since there are a many regional processors, software functions in different function blocks do not necessarily belong to the same regional processor, though they may.

The structure of APT is complicated and since it is not the main object of the investigation, it is described less detailed. More complete descriptions can be found in (ERIC 76). But some basic functions have impact on the analysis and do therefore need to be considered.

Figure 5.2 shows a typical structure of APT. The Subscriber Switching System (SSS) consists of a number of function blocks, which consist of hardware and software.

All subsystems have this organisation and they comprise the functions necessary to establish a connection between a pair of subscribers, either within the exchange or via some other exchange. An AXE can also serve as a transit exchange in which case it only relays calls from one station to another.

The establishment of a connection involves a set of hardware devices. Each belongs to one of the subsystems in the picture above, and a simplified description of the connection routes and the hardware is shown in figure 5.3.

An outgoing call, i.e. a call from a subscriber connected to the exchange, to some other subscriber, connected to some other exchange (for example a long distance call), is described below:

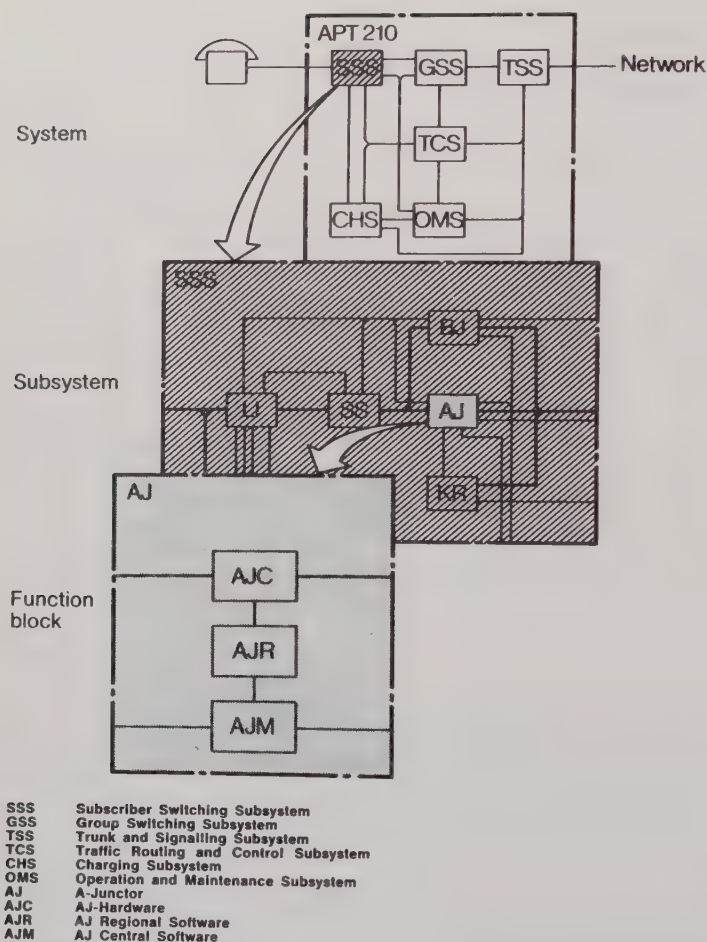


Fig. 5.2 Illustration of functional structure of APT 210

When a subscriber lifts his receiver to make a call, it is discovered by the line circuit in LI. Some category checks are performed and the subscriber is, via SS, connected to an AJ (we assume the telephone to be a dial telephone and not a pushbutton). The digits sent by the subscriber are registered in AJ and stored in RE. When sufficient information is received an outgoing trunk, an OT, is selected, and a path through GS is reserved. Depending on the signalling system, i.e. the mode in which information is sent to the called station, different functions are evoked to send the digits. After completion of the signalling, devices involved in this process are disconnected and a check is made that a speech path has been established. During the conversation devices are supervised by functions in the CL block. The subscriber has, while the activities took place in the system, received appropriate tones that inform about the progress of the call attempt. After completion of the call, equipment is released and charging information is stored.

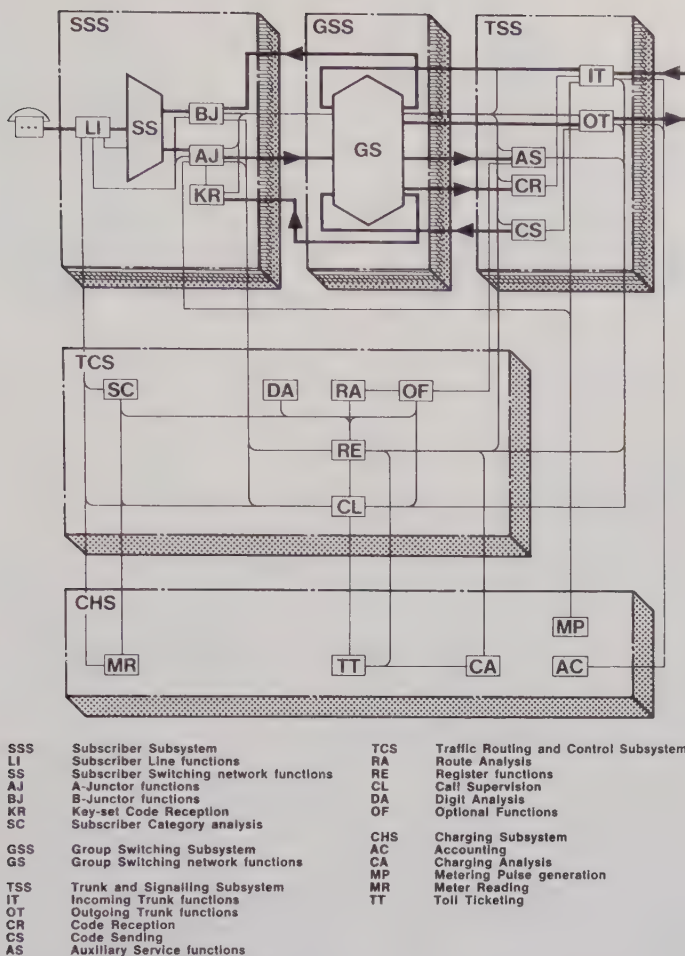


Fig. 5.3 APT 210 Hardware and software functions

In addition to these functions, many activities have to be performed to make the equipment work. Switches have to be activated, current fed to the telephone, tone generators connected and disconnected etc. But it is only some of these functions that are of interest in a performance study, since most of them are independent of the load situation in the system and take equally long time to perform each time they are needed. There is therefore no reason to deal with these functions in detail. The important functions are those performed in software and which depend on processor load.

The data processing part of AXE, APZ, is divided into two principal subsystems: CPS and RPS. There are also two more subsystems, IOS and MAS, but these do not take part in the traffic handling and are not, under normal conditions, limiting the system. The functional structure of APZ is shown in figure 5.4.

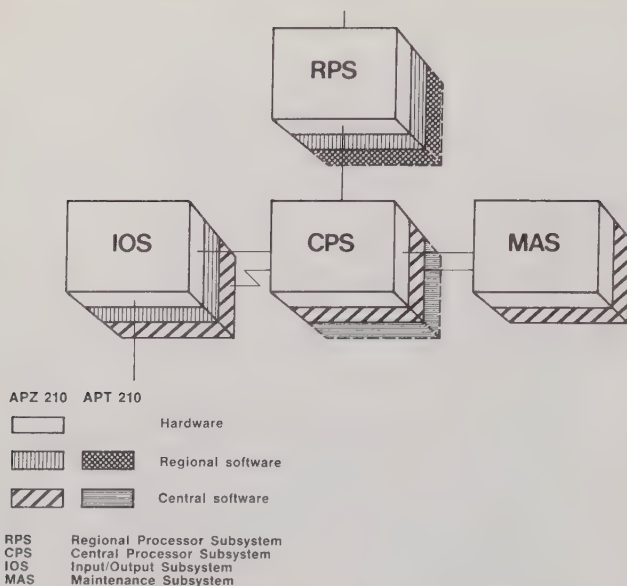


Fig. 5.4 Subsystems of data processing system APZ 210

Note that these subsystems are made up of hardware, central software and regional software in the same way as APT. It is thus the functional structure that is shown in figure 5.4 and not the hardware structure. This is, somewhat simplified, shown in figure 5.5.

The switching hardware is contained in Extension Modules (EM). These are controlled by two processing systems: Regional Processors and a Central processor. All units are for reliability reasons duplicated. Communication between hardware units is established via busses, and there are busses between EMs and RPs and between RPs and CP. The communication between RPs and CP is controlled by a special unit, called the Regional Processor Handler RPH, but its functions will be dealt with later. The software interwork with messages that have unique names and which are sent from one function block to another. It is only at the function block level that communication can take place and this ensures a high degree of data security. Software messages are called signals and if they are exchanged between for example a regional software function block and a central software function block information is sent via the RP-CP bus, but signals that are exchanged between function blocks that are realised entirely as central software do not leave the central processor.

There are a large number of functions that always take a specified time to perform and whose internal behaviour, although sometimes complicated, are of no interest in the simulation. Such functions are for example the scanning of the reed relay matrices that are used to connect circuits to each other. Functions as these are, generally speaking, realised in hardware and do in a simulation only represent some constant delay. This delay may of course be important, but it is not interesting to know how the delay can be further divided. The details of such functions are therefore left out in the presentation.

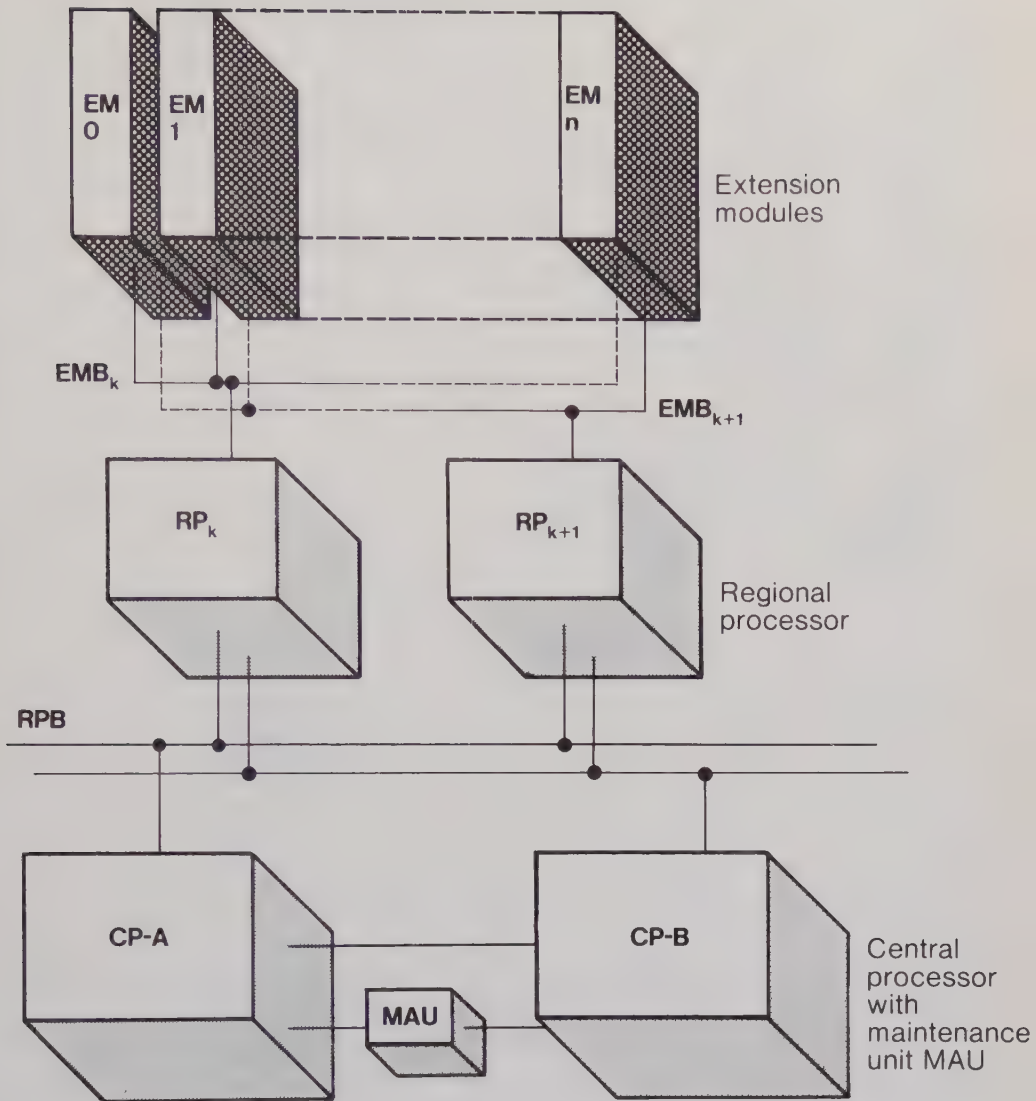


Fig. 5.5 Structure of AXE 10 hardware

The Central Processor

The central processor is organised very much like a general purpose processor. Different units, as ALU, MIG and memory management units, are connected to a common bus, see figure 5.6.

But in addition to these common features, the processor is designed to handle special functions that are convenient in control systems, and furthermore to simplify the special memory organisation which is designed for maximum security. An important part of the memory is the so-called reference memory which acts as a table where addresses are

DS	Data Store
PS	Program Store
RS	Reference Store
CPB	Central Processor Bus
RPB	Regional Processor Bus
MAU	Maintenance Unit
CPU	Central Processing Unit
TCU	Table and Counter Unit
RPH	Regional Processor Handler
ALU	Arithmetic and Logic Unit
BAM	Maintenance Buffer Unit
MIG	Micro Instruction Generator
PCU	Priority Control Unit
TRU	Trace Unit
DSH	Data Store Handler
LIU	Link and Instruction Address Unit
PSH	Program Store Handler
UPM	Updating and Match Unit
RSH	Reference Store Handler
SBU	Shift and Bit Handling Unit

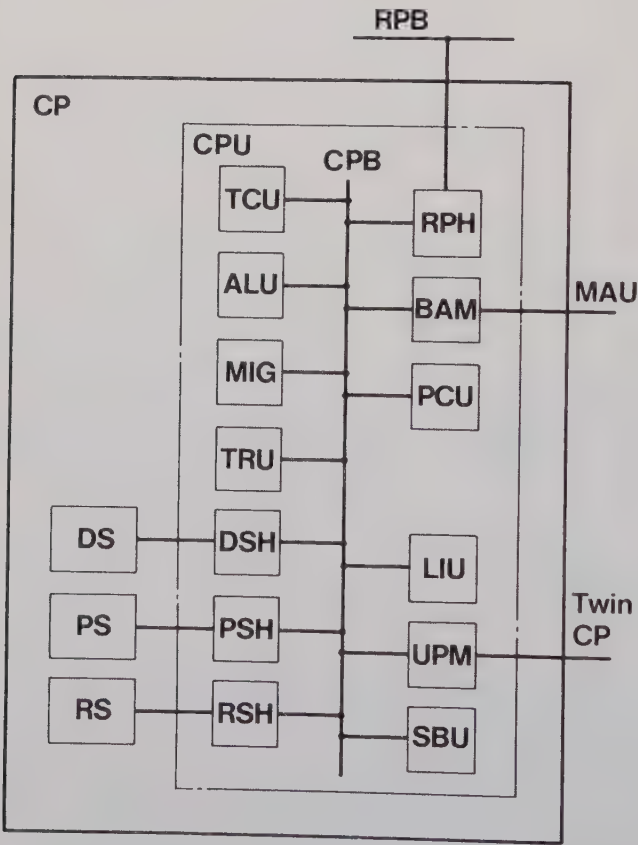


Fig. 5.6 Structure of central processor CP

looked up to avoid unauthorised data access. This extra reference does of course take some time, but since all memory functions are implemented as micro instructions, the extra access does not affect the time delays of interest here.

Of greater importance is the built-in priority structure which enables the system to give priority to tasks that require immediate treatment. The priority structure of the CPU is shown in figure 5.7.

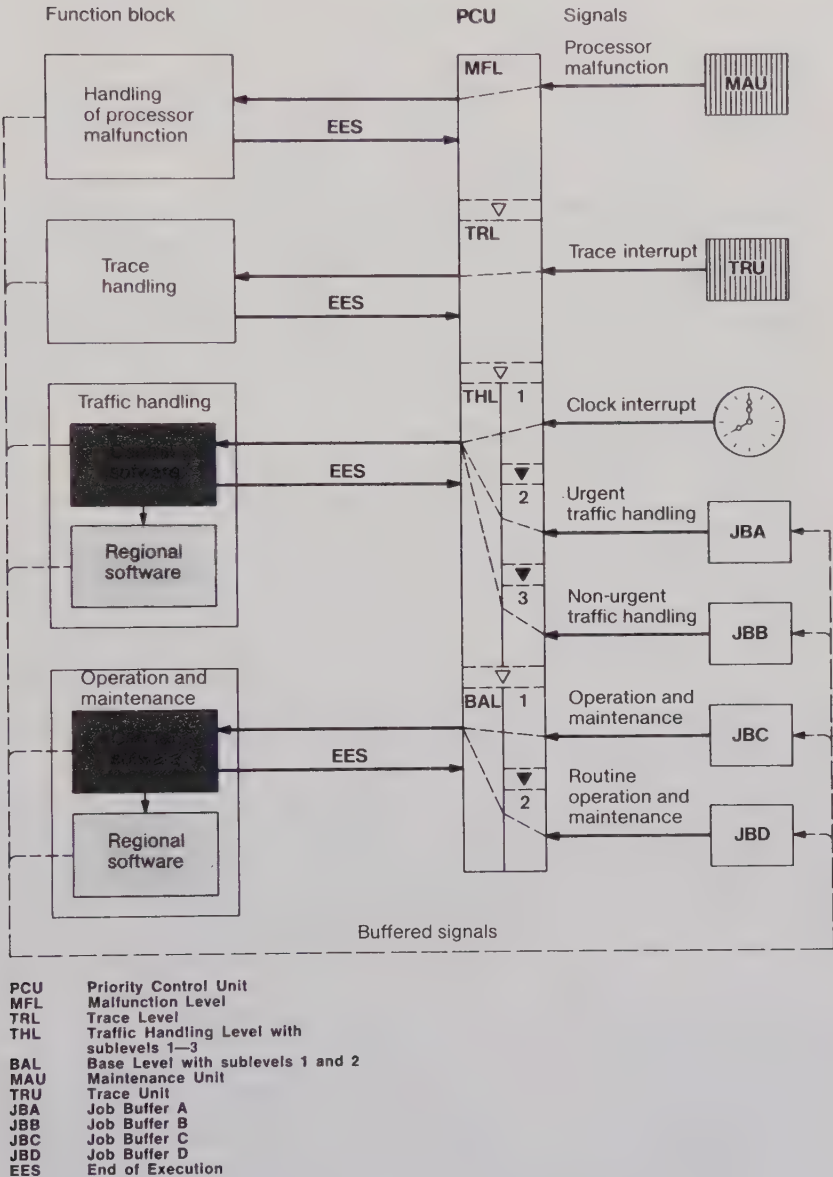


Fig. 5.7 Job scheduling in CP

The different levels have the following interpretation:

The malfunction level MFL has the highest priority. It receives an interrupt when some hardware error is detected.

The trace level TRL is used for program testing. It is possible to trace signals in the system for one reason or another. Tracing is done in hardware and does therefore increase the load only marginally when not activated. This means that checking whether a signal should be traced or not does not under normal conditions affect the system.

The traffic handling level THL is, together with the base level BAL, the normal working levels for the CP. THL is divided into three sublevels, out of which levels two and three have buffers connected, named JBA and JBB. The top level of THL takes care of clock interrupts and scans a job table every 10 ms. JBA and JBB store signals that arrive from function blocks and queue them for processing by the CP. The bulk of signals generated by normal traffic handling enters JBB.

The base level has two sublevels to which JBC and JBD are connected. Service programs are executed at these levels, but do not, under normal circumstances, affect the higher priority levels since signals at BAL are interrupted by signals at THL. The exact behaviour of the priority structure is described in conjunction with the simulation model. Signals flow via the buffers and the CP back to a function block, possibly in an RP, and from there again to a buffer. This flow will be more evident when a signal sequence, that describes how a call is set up, will be described in the simulation model context.

The Regional Processor

Each regional processor is in itself a small computer. It has units connected to a common bus, and program and data stores. See figure 5.8.

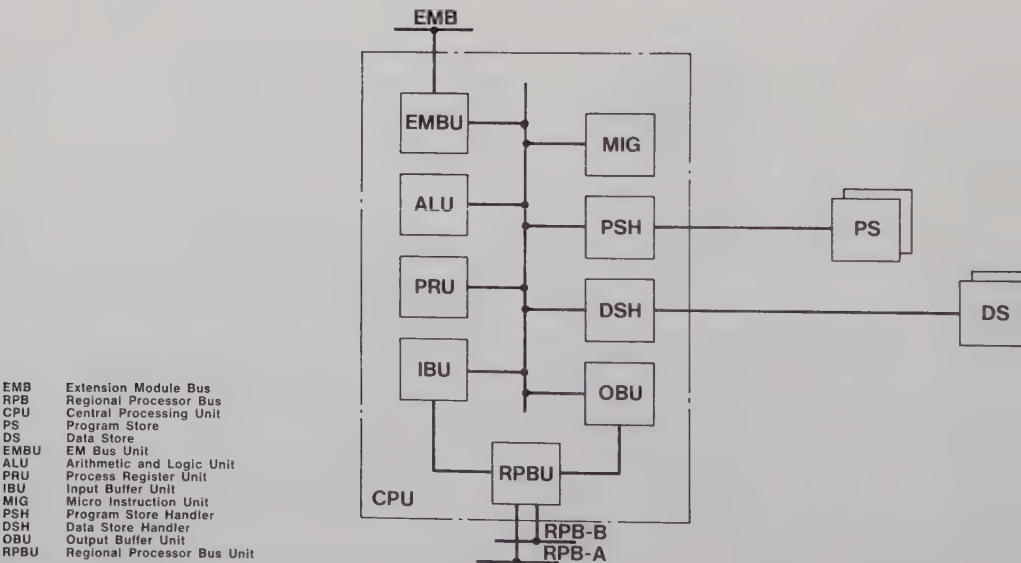


Fig. 5.8 Structure of regional processor RP

A regional processor performs time-consuming tasks of low complexity. It supervises a number of Extension Modules (EM) and communicates with the central processor via the regional processor bus. The communication is handled by a special unit in the CP, the RPH, which will be described below.

The Regional Processor Handler (RPH)

AXE is in a sense a distributed system, but it is the central processor that controls or supervises the system. The regional processors are therefore dependent on the central processor for their work and cannot continue if not looked after by the CP. The CP must for example control the transport of data between RP and CP. This is done by polling. A special device, the RPH, polls the regional processors in turn. The RPH consists of two autonomous parts which share the same bus. One part handles communication RP-CP while the other takes care of the opposite direction CP-RP. RPs are grouped in groups of 32 and such a group has a unique bus at its disposal (see figure 5.5). Each 32 group is further divided into four groups of eight and eight RPs are scanned at a time by the RPH. During an activity performed by the RPH, as for example scanning or data transmission, it is not possible to use the bus for data transport from CP to RP. If such a situation should arise communication CP-RP stops and awaits a free bus. Conflicts of this type should be rare, but may become important in some situations when certain RPs are more likely to be addressed than others.

The scanning mechanism has, as all polling systems, a certain minimum round trip delay. Independent of whether an RP group has anything to deliver or not it takes time for the RPH to scan that group. If the group has something to deliver the scanning time is increased by a term which depends on the amount of data to be transported from RP to CP. Buffers in both RP and CP are limited and it may occur that these become full. In case of a full RP buffer, motion in the RP ceases and starts again when the buffer gets free. A full buffer in CP should occur only under extraordinary circumstances.

More could be said about the communications RP-CP and vice versa, but the details are left until the modelling process.

The I/O functions are generally placed at BAL and do not affect the traffic handling. All I/O is routed via some dedicated RPs, and can in some situations, for example during a dump of charging information, affect the capacity of the bus between RP and CP. Such situations are evidently of interest but can, for reasons that will be discussed later, not be investigated in a simulation of regular type.

5.3 Modelling the AXE system.

A model of the AXE system must, quite obviously, comprise the properties of the original system that are assumed to have impact on the performance measures of interest. The modelling process is mainly to decide

where a proper level of detail is reached and to describe the remaining properties in terms transferable to a simulation program. So described the task is seemingly simple and straightforward. When it comes to action the scene changes.

The capacity of a computer system of type AXE is determined by a combination of hardware and software. Software and hardware descriptions do often differ, and are therefore not easily combined to form a simulation model. This problem is apparent in AXE, where the overall system description is made at a functional level, while it is the hardware components that in the end determine the capacity of the system. Since it is unrealistic to transform the functional description of a system of AXE's complexity to something more suited for simulation, some kind of correspondence between the functional description and the hardware has to be established. This correspondence should be designed in a fashion that simplifies later validation and verification.

There are essentially three groups of models involved in the AXE simulation: models of hardware, software and the outside world. The outside world consists of subscribers and other exchanges. The subscribers can be of many different types and this constitutes a vast modelling problem.

Let us look at the three different kinds of models in turn:

Models of external behaviour

As mentioned already in chapter two, the border between the simulation model and the outside world has to be fixed, and the outside behaviour modelled as generators that generate arrivals or traffic to the system. A subscriber behaviour could be described in the following way: when he has decided to make a call, he lifts the receiver and waits for dial tone. He then dials the digits and waits for ringing tone or busy tone. Depending on the tone received he acts in different ways. If he gets busy tone, he puts the receiver down after a few seconds and tries perhaps again later. If he gets ringing tone he either waits until someone answers or until he decides that no one will answer and therefore puts the receiver down. If he gets answer he will talk for a while, which can differ substantially from one time to another, and then put the receiver down.

The time between for example dialling of the last digit until ringing tone or busy tone depends on how fast the exchange acts. Other delays, as for example time between individual digits, depend on the subscriber. If he knows the number well it is likely that the digits will arrive as fast as the telephone permits him to dial them, while if he is unfamiliar with the number, and perhaps has to look in the directory, it is likely that the digits arrive less fast and perhaps in two or more groups with smaller intervals. Different subscribers wait differently long if they get no answer, and some are inclined to talk longer than others.

To model such a behaviour, it is necessary to keep a record of at least all active, i.e. telephoning, subscribers and to remember in what state of progress they are. Little is known about subscriber behaviour when it comes down to individual performances. Measurements have shown that the pooled arrival process to a fairly large exchange with good approximation can be shown to be close to Poisson (even if this is hard to prove)

(DENN 78). But the same accuracy can not be acquired for individual subscribers.

One question is of course how great impact the time that elapses between for example consecutive digits has on the processor load and the overall performance of the system. Questions like this will be addressed later, but there are others as well: a large exchange can have 40000 or 50000 subscribers connected. If each subscriber generates a traffic of say 0.05 Erlang we end up with an average number of subscribers in conversation of $0.05 \cdot 50000 = 2500$. During a busy hour there can be four to five times as high traffic, which will further increase the problems outlined below. If we want to supervise all these calls and to know when their next activity will take place we have to keep them in the event list, or in some similar list. This list has to be searched each time a new event occurs and the running time of the simulation experiments will be very long. The records of the subscribers will also occupy large memory space, which for some computer systems can be impossible to provide.

The major part of the time between the first attempt to telephone until release of occupied equipment after completion of the call, is spent in conversation. Measurements have shown that this time for some categories of subscribers can be assumed to be exponentially distributed. The signals generated when the subscribers put their handsets down could thus be generated independently and there would be no need to keep a record of the subscribers while they talk. A Poisson generator, with an intensity proportional to the number of set up calls, could generate signals and count down the number of active subscribers. This approach has some dangers however. A generator determines, on basis of the state at one time, the next event of the type to which the generator is associated. If for example there are 2000 subscribers active at a certain time and each subscriber has an intensity of say 1/180 1/s, then the generator determines the next event at a time which is exponentially distributed with mean 180/2000 sec. But if all subscribers were observed and if the rate with which they put their handsets down was constantly updated, it could happen that some subscriber that was not present when the next event was determined, could end his call before the determined time. The accuracy of the approximation depends on the number of active subscribers. 2000 active subscribers correspond to an arrival rate of 11.11 arrivals/sec if the average conversation is 180 sec. The mean time between clear-signals is 0.09 sec during which time on average one arrival occurs. It can of course happen that the change is larger, but the probability that a comparatively large number of arrivals should occur is small for this number of active subscribers.

A model that treats the connection and disconnection phases separately does therefore seem to be a good approximation. The disconnection phase does not involve any subscriber activities and is consequently simple to model with respect to the external behaviour.

The connection phase contains several subscriber activities. The most cumbersome is dialling of digits. Depending on the type of call the subscriber wants to make, i.e. local, long distance etc, there are differently many digits to send. A local call usually consists of five or six digits, while a long distance call has eight, nine or ten or perhaps even more if it is an international call. It is, for reasons that have to do with the soft-

ware structure, not convenient to model the subscribers to be of the same type and to let them be able to make all kinds of calls. Depending on the type of telephone and the kind of call, different activities take place in the exchange. The telephone can for example be either of push-button type or a dial type, and it takes less time to press some buttons than to dial digits in the way we are used to to-day (LIU 80). The model does therefore have to differ between subscribers with pushbutton telephones and regular ones.

An exchange can in principle handle four types of calls: local, incoming, outgoing and transit. Local and outgoing calls are made by subscribers connected to the exchange, while transit and incoming calls come from other exchanges. It is uncommon that an exchange handles all four types. It takes care of either transit traffic only or the other three types and no transit traffic. There are of course special cases where the exchange handles traffic of unusual kinds, but for reasons given already in chapters two and three, it is not feasible to study such systems in a general simulation.

The subscribers are therefore divided into three groups, each representing the subscribers who make incoming, outgoing and local calls. A fourth group represents transit traffic. It may be argued that the subscribers that make local calls and those who make outgoing calls are not independent in this context, but the incomplete information about subscriber behaviour does not leave any option for a dependence model.

Each subscriber group gives on request (the dial tone) its digits to the exchange. But incoming and transit traffic differ in this respect from local and outgoing. Digits concerning incoming and transit traffic are received from some other exchange and arrive without human interference. They can therefore be assumed to arrive with more regular intervals than digits from subscribers directly. The model of the arrival of digits was made as follows: after reception of dial tone, digits arrive with intervals that are samples from the same distribution. No regard is thus paid to any grouping within the digit sequence and digits from incoming and transit calls differ only from local and outgoing what the mean interval between digits is concerned.

All calls except transit calls may find the B-subscriber busy. (The B-subscriber is the called subscriber). They may also find that the B-subscriber does not answer. The points in the chain of activities, that the exchange has to perform to establish a call, where these events are observed, are easy to find. It is therefore easy to find the point at which a call attempt should be ended if the B-subscriber is busy or does not answer. The remaining questions are how long the A-subscriber waits if he does not get any answer and how long it takes for him to put the handset down if B is busy. This latter time may seem to be short, but much happens in the exchange during the few seconds it takes for the subscriber to react. In the model, these times are drawn from some distribution.

Other subscriber behaviour, as for example interrupted calls, retrails, and similar things, are not modelled.

When an exchange wants to communicate with other exchanges, to for example send over digits or charging information, this is done according

to some protocol, i.e. a rule that governs how the information should be sent. Once the exchanges have agreed to communicate, and at some occasions even without this, information arrives in a stream without longer interrupts. This means that the receiving exchange has to have enough capacity and equipment to take care of the arriving information. In this sense there is quite a difference between the communication between two exchanges and a subscriber and an exchange. The communicating exchange, i.e. the one that "our" exchange communicates with, is in spite of this not modelled any different from a normal subscriber. The reason for this is that the time it takes for the other exchange to answer and send digits is indeed what the present investigation tries to find out and it is therefore not possible to estimate how the other exchange performs. It has to be assumed that the times vary around some mean value that can be deduced from the minimum function times of the system. The distribution for this can obviously not be known in advance, but has to be based on experience.

The modelling described above is the most dubious part of the simulation. It contains substantial simplifications and is not easily made more accurate and/or detailed. Its accuracy is limited by our knowledge of human behaviour and the diversity with which man can act. Without running into conclusions one can say that the accuracy of this part sets a limit to the accuracy of the entire simulation provided the other parts can be made equally accurate. It is extremely difficult to estimate the accuracy of the models above, however, and much must be based on intuition.

But it is also possible that the individual subscriber behaviour is of little importance and that much information can be gained about the performance of the system without detailed knowledge about the subscribers. What talks in favour of this is that pooled behaviour, i.e. a behaviour that is the result of many individual and maybe diverse behaviours, has a regularity that changes little if a different mix of individuals is investigated. So even if some talk long while others spend less time telephoning; if some never remember the telephone numbers they use while others always dial them without hesitation, it all adds up to a behaviour that does not change if we change one or a few individual parameters.

These questions are parts of a greater complex that deals with a how sensitive models are to parameter and structure alterations. Chapter seven will show that there are great differences among various properties in this respect.

Models of hardware

Telephone systems in use to-day work with a transmission technique that requires every subscriber to have his unique connection to the exchange. This means in most systems that a subscriber has a pair of wires from the telephone to the exchange and that the total subscriber network, i.e. the wires between subscribers and the exchange, is large and expensive. Other techniques, that concentrates traffic at an earlier stage, are developed but to introduce new transmission techniques in an old

network is complicated. When the traffic is concentrated before it arrives at the exchange new transmission modes are usually introduced as well. The conventional transmission is analogue while the new modes are digital. Digital transmission requires somewhat different circuits in the exchange and works with somewhat different switching schemes.

AXE as described in the previous section is assumed to have analogue connections. This means that each subscriber has a line circuit in the exchange and that these circuits have to be controlled by a device that can observe change of subscriber state. An analogue telephone needs rather large amounts of energy to work as intended. It is made to ring by feeding it with high voltage (approx 100 V) and even when idle it needs about 40 V. The processing system of AXE is a digital device, which cannot handle these high voltages and currents, and some interface between the digital, low-energy processing system and the analogue, high-energy hardware, has to be arranged. This interface can be situated at different places within the exchange, but two principal solutions can be distinguished. There exist in the exchange a device, called the group switching network. This network can be made either analogue or digital, and in the former case the interface is situated between the switching network and the processing system while in the latter the signals from the subscriber are transformed to digital pulses before they arrive at the GSS network which in this case performs digital switching. The AXE system modelled here is assumed to have an analogue network.

A telephone exchange concentrates traffic from the subscribers and spreads it out again. A perfect exchange would at any time be able to connect any subscriber to any other that was not busy, but this would demand the exchange to have an immense switching network, which would be idle most of the time since not all subscribers try to make calls at the same time. The exchange is therefore dimensioned to meet certain demands and can thus in some situations get overloaded. It should in such situations behave gracefully, i.e. it should work as much as it is capable of without attending too much to the extreme situation. The dimensioning has also the effect that some equipment may be scarce and that two subscribers cannot be connected due to lack of connection paths.

When a link through the group network is needed, two entries, one at the input side and one at the output side, are selected and a path is thereafter searched for between these points. If the search fails, two new points are selected and the procedure repeated. This is done a number of times and if no path is found after a maximum number of trials the attempt is considered a failure and the call discharged.

This behaviour is in the simulation model realised by a probabilistic model that has a fixed probability for an unsuccessful attempt. The probability is independent of the number of connections, which of course is an approximation, but since this number varies only marginally for a given load or arrival rate, it is a defensible approximation in this respect.

There are other scarce resources in the system. Devices that receive digits from the subscribers or from other exchanges are provided to a certain number which at times may be too small. If such a situation arises, i.e. if no device can be connected to a subscriber that wants to send digits, no dialling tone is sent and the subscriber has to wait until

he gets tired or a device gets free. This effect is not implemented in the model, but all devices of this type are assumed to exist in sufficient number at any time.

In models of the type considered here, time is important. It is one of the essential goals of the modelling process to estimate the time different actions take and to estimate how they change. Several hardware components are involved in the establishment of a connection between two subscribers. These hardware components have different action times and do therefore have to be treated with more or less care.

The hardware that work the high power parts of the system is comparatively slow. It takes sometimes several tens of ms for a relay to switch from one state to another and this time is, compared with the working speed of the processors and busses, quite long. The switching of relays is also load independent in that it always takes approximately the same time to operate the relay independent of the load situation. They can therefore usually be modelled as constant times and need not be scheduled in the event list.

Most of the hardware that is not processing power is controlled by regional processors. The model assumes the function times of this hardware to be constant and load independent. The simplification is carried even further and it is assumed that the regional processors perform their tasks with constant delays. The argument for this simplification is that the load on the regional processors is much less than the load on the central processor and that the central processor in all situations of interest would be the bottleneck of the system. At first sight this simplification does seem to be acceptable, but later investigations have shown that it may be one of the crucial parts of the modelling. More about this in chapter seven.

The regional processors receive signals from the central processor and process them according to the task scheduling rules of the operating system of the regional processors. Various internal effects may cause the signals to be delayed or reordered in a fashion different from the assumed. These effects are not regarded in the model. If a regional processor finds the output buffer full it stops and waits for the RPH to empty it. This behaviour is not incorporated in the model.

Models of the central processor hardware

Hardware in the central processor executes instructions at the levels discussed in 5.2. MAL and TRL are not involved under normal circumstances. If the system is malfunctioning all models become invalid and there is no reason to run the simulation under such circumstances. MFL and TRL are thus excluded from the model.

The remaining levels, THL and BAL, are modelled as queues served by the same server. Queues are associated with THL 2, THL 3, BAL 1 and BAL 2, and serve as models of JBA, JBB, JBC and JBD, respectively. A special function, evoked every 10 ms, acts as the clock interrupt. At a clock interrupt, a job table is searched and regular events, as for example charging, are executed in CP. Cf. figure 5.7 in 5.2.

The priority structure is Preemptive Resume between different levels but is HOL-NP (Head Of the Line – Non-Preemptive) within the levels, however. The clock interrupts all levels and actions are resumed where they stopped at the interrupt. The load caused by scanning the job table and execution of jobs at this level is described by a hybrid model. The load varies depending on how many of the items in the job table that are due in a specific interval, but apart from more time-consuming jobs, it is unrealistic to model them all in detail. When the job table is searched, only small jobs are executed at THL 1. Larger jobs, that are found to be due in the interval, are transferred to JBB for execution in the same way as traffic handling jobs. As long as the jobs at THL 1 do not differ in structure, i.e. in mix of length and frequency, it is equivalent to model the job table load as an increased arrival rate at THL 2 or THL 3 and to increase load on THL 1. The system performance at a load of 95 % is thus independent of how this load is obtained whether from 10 % job table load and 85 % traffic handling load or from 5 % job table and 90 % traffic handling. Investigations have shown that even if the constant load has different structure it is reasonable to assume the load to come from the same source, as long as the constant part is small.

The central processor hardware is assumed to execute instructions at the same rate independent of the individual instructions. This is an approximation, but since jobs, i.e. programs, usually consist of at least ten and normally twenty to forty instructions taken from the entire set of available instructions it is a good approximation to assume the job to take "number of instructions" * "mean instruction time" time units to execute, (how jobs are queued for execution will be discussed in conjunction with models of the software functions). There is consequently no need to model the internal parts of the central processor as they appear in figure 5.6 in section 5.2. One part is excepted from this however, the regional processor handler.

Models of the Regional Processor Handler (RPH)

The RPH is a part of the CP in the same way as the ALU and the MIG, but it works more in parallel with the rest of the CP than the other components of the CP. Its behaviour is rather complex and involves interrupt signals to the CP controller and intermediate buffering of signals intended for RPs. The speed of the RPH is large compared with other components of the system and especially with delays encountered in hardware switching functions. The question of how detailed it should be modelled is connected to how the communication between RPs and the CP is modelled as a whole.

Signals exchanged between CP and RPs carry information about a call or some other information, for example data to be written on some I/O device. Different signals have differently large data fields, and do therefore require more or less time on the bus. A detailed model would for each call have to picture the exact size of every signal sent on the RP bus. It would also keep track of the order in which signals were sent and in what state the input and output parts of the RPH were, in order to be able to delay signals if they were sent and received on the same bus. The output part of the RPH (i.e. the part that handles communication from

CP to RP), has a register in which the CP controller puts signals intended for some RP. It may occur however, that this register is full. In such a case the controller stores the signal in a buffer (JBR) from which the RPH takes them when it gets ready. This intermediate buffering is avoided under normal circumstances since it takes some time to perform. But an interesting question could perhaps be: can the buffer JBR get full. To answer this question would require the model to involve all the details described above and perhaps also a more detailed RP description.

The RPH was, as a compromise, modelled as two almost independent components, with a limited interwork. The input part of RPH picks signals from a number of queues that are shared by the RPs. It is assumed that most hardware devices of a specific type are connected to RPs on the same group of 32 and they do consequently share the same bus. The RP buses are polled, and if some RP on the bus has anything to deliver a scan takes somewhat longer time to perform. It is also assumed that all buses are scanned independent of whether there are any RPs connected to them or not.

The output part of RPH takes signals from a queue which models JBR and transmits them to an RP. The transmission is assumed to require a time that is a sample from an adequate distribution. No individual transmission times are thus maintained. The queue is assumed to be infinitely large and nothing happens if the queue size exceeds some specific value. The RPs accept the signals promptly and no delay is therefore caused by the RPs in this respect.

Models of software functions

Functions in AXE are realised either as software or hardware, and the former can be either central or regional software. Each function belongs to a function block, which can consist of combinations of hardware and software. Function blocks communicate via signals, that may be either software or hardware signals. The regional processors and their software functions are, as already mentioned, modelled as load independent delays and are not treated in detail.

When a call attempt is discovered by a hardware device supervised by some regional processor a sequence of activities is started. This can be viewed as the exchange of signals between different function blocks. The establishment of a connection between two subscribers involves a specified sequence of tasks, and some of these tasks are realised as software. Different types of calls, i.e. incoming, local and so forth, have different sequences, and a model of the software system has to comprise all software functions. Load on the central processor depends on the number of calls treated by the system at that time and on in which state of progress these calls are. Some activities, like for example digit analysis, require more processing capacity than others, as for example supervision of an established call. A model of the software function has to be able to generate load on the central processor in proportion to the number of calls and their states. This means that a model of the calls has to be built and that this model must allow the calls to flow through the sys-

tem in accordance with the sequences that define the necessary activities.

The sequence that defines the function blocks activated by a local call is in part shown in the figure below. It shows the beginning of the sequence, and the arrows indicate software signals while the boxes indicate function blocks which are passed as the "call" propagates through the necessary activities.

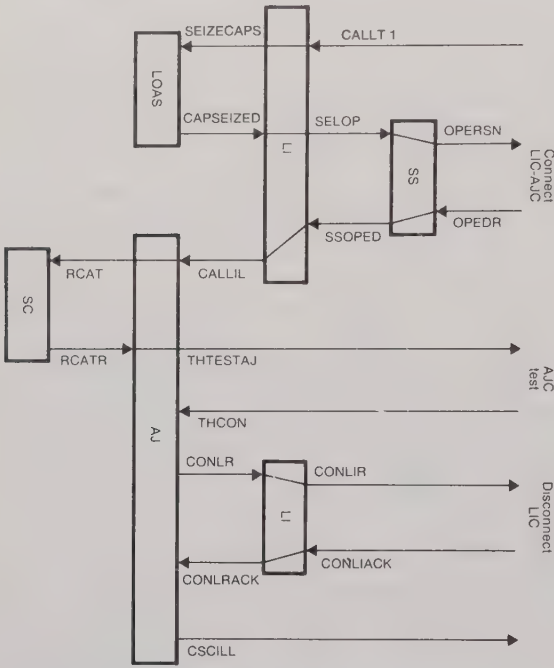


Fig. 5.9 Signal and function block sequence

The boxes in the figure above represent execution of central software. The abbreviations LI, LOAS, SS and so forth are names of different function blocks which take care of specific software and hardware functions. The function block LI for example consists of functions which are used to connect the subscriber to the exchange (Line-Interface). The sequence shown here does not contain any hardware switching functions, but these are activated by regional processors as indicated by the arrows that point upwards and out of the figure. Connect LIC-AJC for example means that a signal is sent to a regional processor, which controls the Line Interface Circuits and the A-subscriber Junction Circuits, to operate these functions.

If each call could be made to follow a path as indicated in figure 5.9, and if the activation times of the switching hardware and the regional processors could be adequately modelled, then the central processor load could be derived (in a simulation) from the joint effect of all active calls. The information required for this is the number of instructions each central software task contains and the activation times of hardware functions controlled by RPs. This information can be found in the source listings of the software and in the descriptions of the hardware.

The software is, with the above considerations, modelled as a sequence of jobs that have to be processed by the central processor hardware and which are delayed by excursions to switching hardware and regional processors. A call is created by a generator and propagates through different function blocks in which it encounters different delays, i.e. jobs that consist of differently many instructions. At some points in the function block sequence regional processing is required in which case the call is transferred to a regional processor which sends it back when finished. The only problem with this approach is that it is hard to find the central processor since it ought to be at many places at the same time to process calls that are in different function blocks. How this dilemma is resolved will be shown in the next section where the program structure is discussed.

Earlier was mentioned that the regional processors were modelled as a constant delay. This delay differs however with the kind of activity the particular RP controls. Some activities are relatively fast, as for example connection of two devices, while others are more slow, as an extreme example, which deals with test of a relay twice with 180 ms interval, shows. This means that all RPs cannot be treated in the same way. A specific RP activity is therefore modelled as a constant delay characteristic to that function. A distributed delay models the fact that a signal enters some time during an RPs primary interval and consequently is not discovered until the next time the RP comes back to the buffer and looks for new signals.

When a call arrives at a function block and is to be treated there by some central software, it is assigned a number which tells how long, in number of instructions, the action takes. With this number it enters the appropriate CP queue. When it, perhaps after some waiting, arrives at the CP it is delayed in proportion to the number of instructions it carries. All instructions are consequently assumed to be equally long and one job is assumed not to be affected by the previous one. This is an approximation since the CP has instruction prefetch, which sometimes succeeds and sometimes does not, depending on the previous instruction.

The software signals, with which the CP is made aware of jobs to be processed, take time to transfer from one block to another. These times are added to the execution times already when a call gets its number at the function block it currently visits. It does also take some time to transfer signals that arrive from RPs to the appropriate CP queue. These times are added to the job table level as a constant load since the transfer is done at a higher level and therefore from the THL looks as a job table delay.

5.4 A Simulation Model

It is hard to describe a simulation model without listing programs, but it is on the other hand impossible and unnecessary to describe every statement of a large program in detail. The major part of a simulation pro-

gram is regular programming which is understood by one and all (who knows programming), but there are parts that are of interest in a more general sense. The problem is how these interesting parts could be discussed without giving the complete program. In the case of the AXE simulation this may be possible (it is indeed in the eye of the reader), because large parts of the simulation program have in principle the same structure, and if the function of one part is understood (with the help of an example, which is program listings), it becomes easy to imagine the rest and thereby to create a frame for other examples. For some parts the model implementation is obvious once the model is explained and it is in these cases unnecessary to describe the program. This section deals with the more interesting parts of a fairly large program and will discuss the implementation in general terms.

A program used actively does often exist in several versions and the AXE simulation is no exception. The version discussed here has the highest fidelity of all versions that have emanated from the same original and has been used to produce results without any special modifications, which is the case with all other versions.

A language has to be chosen to implement the program. There are many considerations involved in the selection and it is not only the relative benefits of a particular language that make the answer. The regular program problems as portability, readability, efficiency and availability are also involved. A simulation program implementing an AXE model was likely to be large, and it was therefore essential that the programming could be done in a language that supported the models and the descriptions, and which was more likely to produce a correct program than others.

By this and other reasons, SIMULA was chosen. SIMULA violates some of the requirements listed above, but does also meet others more readily than any other known language. SIMULA is not generally available and is not known by as many as for example FORTRAN, but it contains elements that simplify some of the processes involved in simulation programming. It is based on the process approach (see section 2.3.2) and has facilities for list handling, data structures and scheduling of events in a simulation. A thorough description can be found in (BIRT 73).

The problems related in 5.3 become even more apparent when the models are to be implemented. Simulations based on the process approach must describe processes as modules in the simulation program. There are, in AXE, some obvious picks in this respect, for example the central processor and the regional picks processor handler. The central processor executes instructions attached to specific jobs at some level and continues to do so "for ever". It is obvious that the central processor should be implemented as the hardware structure, and that no other decisions than to pick the next job to be processed should be put in the central processor. But this approach does not easily comprise the sequence a call has to go through when it enters the exchange.

There have consequently to be some parts of the program that illustrate the signal sequences and which could be used by the call as it propagates through the system. For illustration of this, look at the figure below.

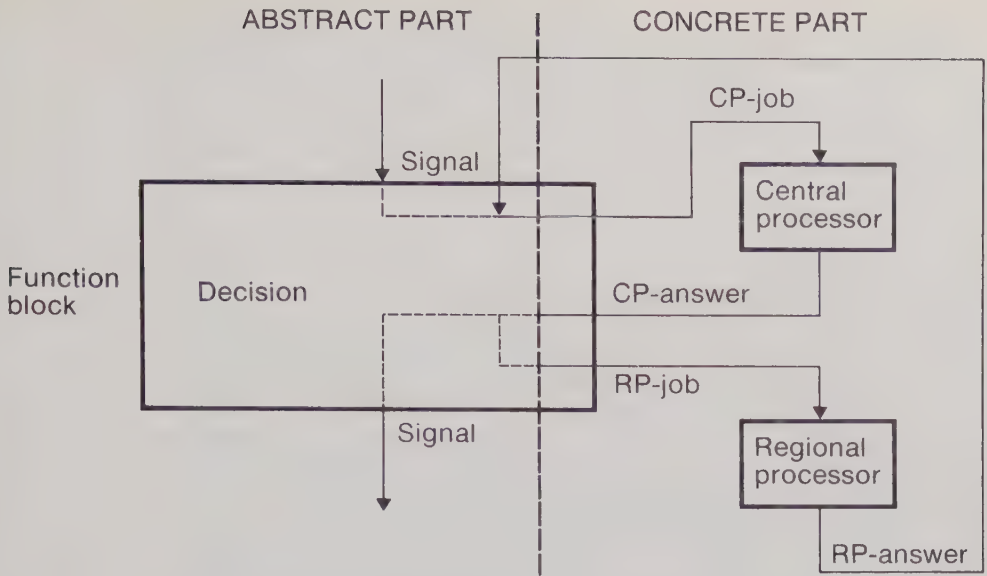


Fig. 5.10 Model communication

The left side of the figure illustrates the functions of the AXE system, while the right illustrates the processing hardware. The functions are described by the function blocks. When a function is performed by a block it involves actions in either switching hardware or in processing hardware. A function block can receive certain signals and can emit others. To establish a connection between two subscribers does therefore mean to go through a sequence of signals which are exchanged between different function blocks. An approach would thus be to write a program segment which was made to recognise a set of signals and which answered by emitting the appropriate answer determined by the sequence it was involved in. By writing these function blocks one would get a picture of the functional behaviour of the exchange, but it would not create any load on the processing hardware.

To illustrate how calls enter the system and how they are treated by the hardware, a number of classes, each describing a function block, were written. These classes were made able to understand, or recognise, three different kinds of input: messages from other function blocks, messages from regional processors and messages from the central processor. They were also made able to emit two kinds of output: messages to the central processor and messages to regional processors.

These messages were made as subclasses to a class MESSAGE in the following way:

```
MESSAGE CLASS SIGNAL;
MESSAGE CLASS CP-JOB;
MESSAGE CLASS CP-ANSWER;
MESSAGE CLASS RP-JOB;
MESSAGE CLASS RP-ANSWER;
```

Without going into detail it is assumed that these classes carry information sufficient for the receiving unit. The only important parameter in this context is a text variable which is used to carry the name of the message. Signals that are sent between two function blocks, have a unique name, a mnemonic, and this name can be used to identify the signals throughout the simulation. The beginning of the sequence, which sets up a call, is shown in figure 5.9 above.

When for example the signal CALLT1 enters the function block LI some actions take place in the central software. This is in the simulation program arranged by a transformation of the incoming signal to a CP-JOB with the same name, which is sent to the class illustrating the central processor, as shown in figure 5.10. The CP-JOB is queued in the processor for processing and when finished it is again transformed, this time to a CP-ANSWER, and sent back to the function block from which it came. A CP-ANSWER can, depending on where in the sequence it is received, give rise to either a signal to some other function block or an RP-JOB. The outcome is determined by the function block, and if it is an RP-JOB the appropriate RP is selected and activated. (The original signal name is kept until the CP-ANSWER is received and thereafter changed, either to a new signal or to an RP-JOB).

The RP-JOB has, on its way to the RP, to pass the RPH. The RP-JOB is therefore put in a queue from which the output part of the RPH takes jobs for processing, i.e. transmission to the appropriate RP. The RPH works as an G/G/1 system with a service time distribution which describes the length of different RP-JOBs. The RPs are assumed to be infinite servers and no queueing take place in these. The service times of these servers are constants which illustrate the hardware activity controlled by the specific RP and which is given by the name of the RP-JOB. In addition to the constant a random delay is selected that illustrates the time it takes for the RP to discover the arrived RP-JOB.

The answer to the RP-JOB is determined by the function block that sent the RP-JOB. This answer is transported by the input part of the RPH, which is a polling system. The function block receives the RP-ANSWER from the RPH, which it recognises and transforms to a CP-JOB with the same name.

If the messages are attributed with some extra information, as for example the length of an CP-JOB, the destination of an RP-JOB, and the call to which the information is attached, it is possible to program a section of a function block which is shown in figure 5.11.

The sequence of function blocks shown here illustrates actions taken when a local call enters the system, but similar sequences apply to incoming, outgoing and transit calls as well. Actions taken for maintenance and operation follow similar sequences, but other function blocks are involved. This means, that the structure with a software part, which incorporates the function blocks and thereby the functional description, can be used for all levels of program execution in the CP, provided the communication is established with MESSAGES as earlier defined. The approach has other advantages, but a discussion around them will be postponed until later.


```

FUNCTIONBLOCK CLASS AJ;

BEGIN
  WHILE TRUE DO
  BEGIN
    PASSIVATE;
    MESSAGE-POINTER := MESSAGE-BUFFER.FIRST;
    TYPE-OF-MESSAGE := MESSAGE-POINTER.MESSAGENAME;
    IF MESSAGE-POINTER IS SIGNAL
    THEN
    BEGIN
      IF TYPE-OF-MESSAGE = "CALLIL "
      THEN
        NEW CP-JOB("CALLIL ",MESSAGE-POINTER.SUBSCRIBER,INSTRUCTIONS)
          .INTO(CENTRAL-PROCESSOR-POINTER.JBB)
      ELSE
        IF TYPE-OF-MESSAGE = "RCATR "
        THEN
          NEW CP-JOB("RCATR ",MESSAGE-POINTER.SUBSCRIBER,INSTRUCTIONS)
            .INTO(CENTRAL-PROCESSOR-POINTER.JBB)
        ELSE
          IF TYPE-OF-MESSAGE = "THCON "
          THEN
            NEW CP-JOB("THCON ",MESSAGE-POINTER.SUBSCRIBER,INSTRUCTIONS)
              .INTO(CENTRAL-PROCESSOR-POINTER.JBB)
          ELSE
            IF TYPE-OF-MESSAGE = "CONLRACK "
            THEN
              NEW CP-JOB("CONLRACK ",MESSAGE-POINTER.SUBSCRIBER,INSTRUCTIONS)
                .INTO(CENTRAL-PROCESSOR-POINTER.JBB)
            END
          ELSE
            IF MESSAGE-POINTER IS CP-ANSWER
            THEN
            BEGIN
              IF TYPE-OF-MESSAGE = "CALLIL "
              THEN
                NEW SIGNAL("RCAT ",MESSAGE-POINTER.SUBSCRIBER)
                  .INTO(SC-POINTERS.MESSAGEBUFFER)
              ELSE
                IF TYPE-OF-MESSAGE = "RCATR "
                THEN
                  NEW RP-JOB("THTESTAJ ",MESSAGE-POINTER.SUBSCRIBER,RP-NUMBER)
                    .INTO(RP.BUFFER)
                ELSE
                  IF TYPE-OF-MESSAGE = "THCON "
                  THEN
                    NEW SIGNAL("CONLR ",MESSAGE-POINTER.SUBSCRIBER)
                      .INTO(LI-POINTERS.MESSAGEBUFFER)
                ELSE
                  IF TYPE-OF-MESSAGE = "CONLRACK "
                  THEN
                    NEW RP-JOB("CSCILL ",MESSAGE-POINTER.SUBSCRIBER,RP-NUMBER)
                      .INTO(RP.BUFFER)
                  END
                END
              END
            END
            IF MESSAGE-POINTER IS RP-ANSWER
            THEN
              IF TYPE-OF-MESSAGE = "THCON "
              THEN
                NEW CP-JOB("THCON ",MESSAGE-POINTER.SUBSCRIBER,INSTRUCTIONS)
                  .INTO(CENTRAL-PROCESSOR-POINTER.JBB); END;
            END*** AJ ***;
          END
        END
      END
    END
  END

```

Fig. 5.11 Program section showing function block structure

The different types of calls follow different sequences of jobs, since they need different kinds of attention. An outgoing call for example has to be assigned an outgoing trunk, which is not the case for a call that stays within the exchange, as a local call. Since it is a unique sequence of signals that determines the actions to be taken, it is only necessary to specify the originating signal. This is conveniently done in the generator that generates calls to the exchange. The individual call streams are assumed to be independent, and four different generators, one for each call type described above, were therefore programmed. There are, within each group, subgroups that may differ in some respects. A local call for example can arrive from either a pushbutton telephone or a dial telephone, but the signal names attached to the treatment of these subgroups do not differ and it was therefore not motivated to create special generators for these calls.

The calls were not, for efficiency reasons, kept during conversation. This would have meant thousands of calls (regional processors to be precise) in the event list of the simulation and a scan of the major part of them each time a new event was to be scheduled. This would considerably slow down the simulation and would also create problems with memory space (something that happened in spite of calls not being kept during conversation). But the disconnection of subscribers creates load on the processors and does therefore have to be modelled. This was done by writing special generators for the event "end of conversation", and to let them run with a speed proportional to the number of calls in conversation. This assumes disconnection to be independent of connection or at least to have a minor significance.

A number of conditions have to be met for this to be true. The time a conversation lasts has to be "long" compared with the time it takes to set up the call. In such a case are bursts of arrivals smoothed out by the randomness of the conversations and no similar burst can be expected for disconnections. It is also necessary to require that no measurements extend over the conversation period. Since a call cannot be traced from origin to disconnection, it is not possible to measure for example correlation between connection and disconnection times. When a call is disconnected, the number of conversations has to be counted down to lower the rate with which new disconnections appear. If all subscribers belong to the same group, this is usually no problem, but difficulties can arise if the subscribers for some reason are divided into groups (they can for example share the same device).

This does not cause any problems in models that illustrate telephone traffic since the conversation here lasts for several minutes on average, which is long compared with the set up times which are in magnitude of a few seconds. But for other types of traffic, as for example data traffic, where "conversation" times may be only a few seconds, problems may arise.

The solution to have several generators of disconnection activities can therefore be defended in this context.

The signal sequence shown in figure 5.12 is somewhat simplified. It is in this assumed that the sequence is continuous and that one signal is the direct result of the previous one. This is not always true however, and an example is shown below.

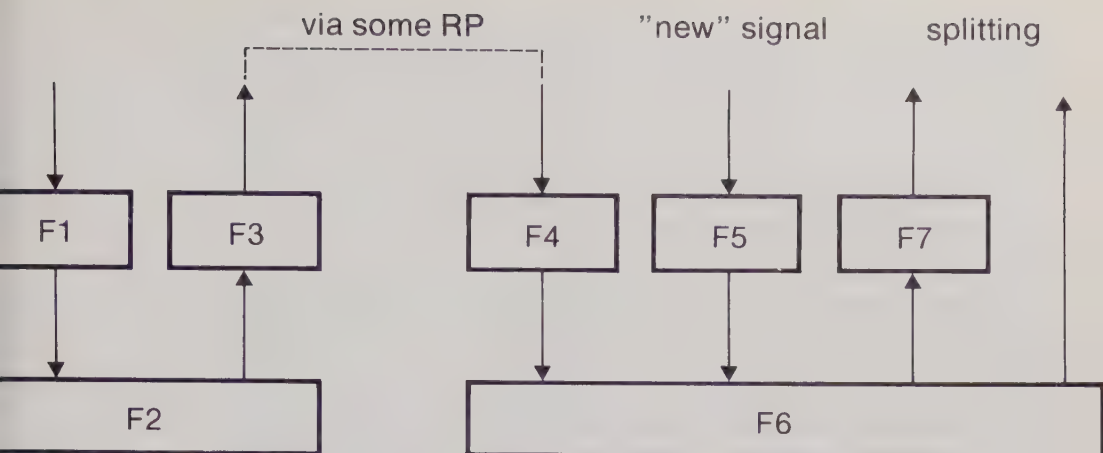


Fig. 5.12 Signal sequence anomalies

There are two irregularities in this figure. The first is that signals at some points split into two and the second that signals sometimes appear as "new", i.e. without being the result of a previous signal. These anomalies create problems in the simulation which are briefly discussed below.

A call is assumed to propagate through the system and to encounter delays in hardware, both processing and switching hardware. The state of an individual call is remembered by itself with the aid of a pointer which points at the function block to which the call is currently attached. If the signal sequence splits, and creates two parallel branches, the execution of tasks in one of the branches has no call to remember the state. If we were interested in only the processor load it would be tempting to "straighten out" the signal sequence and to put the parallel branch at some place within the other. This would not affect the total processor load (since all signals would eventually be executed), and it would in much simplify the programming. This approach was, luckily enough, as will be shown later, not chosen. Instead a dummy or phantom call was created that followed the parallel branch but which was not counted or measured. This phantom call did of course have to have all the attributes of the call it originated from and this parameter transmission was unlucky since it introduced a source of error in the program.

The other anomaly, that signals appear as new, can be handled in the same way as the splitting process. Since signals appear independent of each other, the same call cannot be made to follow all sequences that indeed are the result of the same original call. The original call is therefore copied to all the calls that are used in the sequence and only the last is allowed to continue after the independent arrivals have occurred. A typical example of this situation is the arrival of digits from the subscriber. Digits arrive in sequence, but it may happen that the next arrives before the previous has been processed by the system. A single call can therefore not be used to follow all the digits through the system. When the final digit has arrived and been processed only one call should remain. This is arranged by letting the function blocks involved in digit analysis stop those calls not meant to continue. This is easily accomplished by not emitting any new signal as the answer is received.

Splitting of signals and simultaneous appearance of signals make it hard to estimate the number of calls, real and phantom, in the system. The requirements for memory are consequently also hard to estimate. Another parameter in this context is the time between arrival of individual digits. Longer times will make a call stay longer in the connection phase and thus to occupy memory for a longer period of time. Dial telephones differ from pushbutton telephones in this respect since it is simpler to dial a number on the buttons than to dial it on the dial.

The talk about memory requirements is a consequence of the absence of virtual memory on most machines with SIMULA compilers. UNIVAC 1100 for example, on which the present program was developed and run, does not provide virtual memory facilities and the memory is therefore quite easy to fill with objects. If the system works near the upper limit of the available memory space, a large number of garbage collections is created which take a considerable part of the total execution time. There are thus many reasons to discuss allowable simplifications that lessen the requirements for memory. And even if the machine has virtual memory, a long event list will slow down the execution which may be a severe drawback.

The clock interrupt was programmed as a generator, which changed a variable in the CP every 10 ms and which made the CP aware of the change.

So far there were no problems with the implementation of the model as described in 5.3. The basic structure was clear and to transform it to program code was indeed a gigantic task, but straightforward. Some things remained and these would turn out to be almost as complicated as the rest of the programming taken together, inspite of their much lesser size. Two of these arose because the program should be easy to change or amend and one because of the complicated functions to be implemented.

Each time a clock interrupt occurs, a job table is searched. The appearance of this job table depends on the functions implemented in the particular exchange. If a large number of functions is evoked via the job table, substantial time will be spent at THL1 which will lessen the time available for the other levels. But since not all jobs in the job table are due in all intervals, it is hard to estimate the load on THL1 and how it changes with time. It is therefore necessary to picture the job table and to illustrate the jobs in it. But if different exchanges should be investigated with the same program, it would have to be fairly easy to change the content of the job table. To accomplish this, a structure similar to the "real", i.e. the one in the AXE exchange, was built. This structure became complicated and will not be described here. The reason for the complexity will be discussed after a description of the other "messy" programming problems: the load regulation and the measurements.

Measurements had to be implemented as carefully as the job table. To add new routines for measurement had to be simple and this created substantial problems in the implementation. Measurements will be discussed more in detail in 5.6, and only the programming implications will be dealt with here. Measurements in simulation programs are of two major types: those connected to an event which happens as a result of the execution, without prior knowledge to the modeller, and those which take place with intervals specified by the modeller. These intervals can

be constant or taken from some distribution. Support for these measurements was incorporated in the program, but the interface towards the user did not become clean. It is as difficult to enter a new measurement as it is to add new functions to the job table. The reason for this will be discussed below.

Load regulation is needed in any computer system which acts in an environment similar to AXE's and which has similar demands imposed on it. The environment of AXE appears to be stochastic and does sometimes require more than the system can provide. The system should, in spite of this, work with useful work, and not attend too much to the overload situation. It should also give customers a fair degree of service and control the situation until things become better. A load regulation is needed to meet these requirements. The load regulation should act on the current status of the system, and its actions can therefore not be specified in advance. It has for example to know the load of the system and how many calls that are currently being processed. These figures must be collected in the hardware and information does therefore have to be exchanged between the function block that describes the load regulation and the hardware, as for instance the central processor.

The simulation was based on the function blocks and the hardware and on the fact that these parts did not exchange information in any other way than by means of signals. The load regulation did not fit into this structure and the programming did therefore become complicated.

This reason is indeed true for all three examples given above. To implement a function, that does not follow the intentions and the structure of the original system, creates problems and programming gets complicated. And this does not depend on the way in which the program is constructed, but on the information access needed by the functions. If the program contains any errors, it is likely that they can be found in the parts that implement the functions described above. An approach that would have in much improved the structure of the program would have been to transform the load regulation, the measurements and the job table investigation to a function block structure. This is obvious in retrospect, but was not thought of at the time.

5.5 Verification and Validation

The simulation program described in 5.4 had to be verified and validated as described in chapter 4. The possibilities to validate the model were limited, since only a few exchanges had actually been built at that time and since the measurements performed on these were neither well documented nor especially extensive.

Verification is not affected by the existence of the "real" system, but validation can, as already seen, not be successfully conducted without a target system. The AXE program and model have been verified and validated in accordance with the ideas outlined in chapter four. The main parts of these processes will be described below, and some general reflections will be intermingled.

5.5.1 Verification

In chapter four, two ad hoc notations for verification were introduced. They were called verification type I and II, and differed mainly in the respect that the latter had to comprise measurements of random variables defined in the model. The possible set of such verifications is large and a limited number has to be selected. The selection is of course arbitrary but has been guided by the the existence of analytical results with which to compare and by the interest paid to that particular measure. The ease, with which different measures can be compared differs, and it is sometimes only possible to compare means, while in other cases it can be possible to compare distributions. A mean value can certainly compare well without the model being correct, and this type of validation is consequently less valuable than a validation in which higher moments are involved as well. Higher moments are on the other hand more difficult to measure and require more extensive test runs than mean values. A compromise had in most cases to be made, which will be pointed out in each case.

Verification type I.

The most extensive version of the program consists currently of approximately 8000 SIMULA statements. Out of these, approximately 6000 describe the function blocks while the rest describe other parts of the simulation, as hardware functions, generators and so forth. A minor part is involved in input and output and measurements.

Already when the simulation was constructed, attention was paid to the verification problem. The result was that a number of "unnecessary" features was maintained in the program which could be used to simplify the verification. One of these features is the use of TEXT variables for storing the signal names as they appear in the signal sequences (see figure 5.9 in 5.2). The message components, i e SIGNAL, CP-JOB, CP-ANSWER, RP-JOB and RP-ANSWER, contain a variable of type TEXT which is used for storing the name of the signal from which the message emanates. All messages have to pass a function block, and it becomes easy to check the appearance of the appropriate sequence if the name of the message is put out to, for example, a line printer each time a message enters or leaves a function block. If more than one call were let through the system, sequences would mix, and it would be impossible to follow the sequence resulting from a single call. The call generators were therefore changed to emit only one call which was followed through the system.

The load on the central processor is a relevant measure which depends on the arrival rate and the service requests. The latter are assigned to the CP-JOBs which carry them to the central processor. It can be seen in the short program segments in section 5.4 how the processing time of each job appears as a number in the generation of a new CP-JOB. This figure has to be correct to produce a correct processor load when the aggregated effect is measured in the central processor. When a single call was let through the system, a series of jobs arrived at the central processor. Each of these carried a service time which was put out to an external medium. The lists thus produced were compared with lists obtained from the system programs and the correctness of the assigned service times could thereby be verified.

A call can encounter a large number of different states as it enters the system. The actual outcome for a specific call is in the simulation governed by stochastic distributions, and can thus not be determined in advance. To overcome this, input data was changed in order to make the simulation follow a predetermined path. The likelihood for the B-subscriber to be busy could for example be set to its two extremes, one and zero, which creates a path that could be determined in advance. The intermediate values of such a probability is difficult to verify since the actual outcome always is either true or false. It is only with a large number of attempts that the probability can be verified. This problem is connected to the problem of how the system behaves when there is a large number of concurrent calls active at the same time. The technique applied above can not be used to verify the simulation in this respect, but this has to be done with a type II verification.

Many combinations of input values for the stochastic distributions, which governed the fate of an individual call, were tested and the results were compared with the expected signal sequences.

A call that has entered the exchange must, sooner or later, be released. This means that at the end of a test run there has to be a correspondence between the number of arrived calls, the number of released calls and the number of calls still in the system. The first two of these numbers are simple to count, but the last is somewhat more difficult. A signal sequence can, as already described, split into two or more, and this was in the simulation realised by creating dummy or phantom calls that existed simultaneously. If these were counted, a too large number of active calls would be found. The only way to count the actual number was to count up and down variables when such an event occurred that meant a change of these numbers. This method is not satisfactory, since there very well may remain situations where the variables are not changed although they should be, and where the updating is wrong. The estimated number of calls was at any rate compared with the expected and made to agree.

The signal sequences are at some points interrupted only to continue with the arrival of a "new" signal created by a generator. This is for example the case for digits which arrive independently. The verification contained checks of the existence of the proper sequences and of the number of arrived digits, which could differ depending on the type of call. Checks similar to the above were also performed for released calls which had their own generator and which consequently had to be verified separately.

The verifications described so far have dealt mainly with the functional parts of the system, i.e. the function blocks and the signal sequences. The hardware parts, which are smaller, but in a sense more complicated, could not be verified by following a job on its way through the system. The central processor for instance comprises a priority structure which is rather complicated. This structure can act incorrectly without the simulation acting noticeably incorrect. (The simulation did in fact for a long time contain an error in this structure that was not observed.) The behaviour of this priority structure was to a certain extent verified by tracing the central processor as it worked with jobs arriving from the

function blocks. This verification was insufficient, however, and extensive type II verifications had to be conducted.

A most cumbersome part of the simulation was the load regulation. The load regulation should, when certain levels were reached in the system, limit the number of calls admitted into the processing system. To obtain a good algorithm for this purpose, a substantial amount of data had to be collected and remembered. This collection and updating were organised from the job table and demanded declaration of a number of "unnecessary" variables and access of data in a way that violated the program structure. The reason for this somewhat "messy" programming was the inconsistency of the description of the load regulation. The updating of each individual variable, and the function of isolated parts of the mechanism, could be verified, but it was not possible to "prove" anything about the joint effects of the load regulation. If ten to fifteen variables are to be updated, and different actions taken depending on the state of these variables, many different situations can arise and it is impossible to check, and perhaps even to think of, them all. Another question is of course whether the implementation works in the "real" system.

The function of the load regulation is one of the weak spots of the simulation, and this and other reasons indicate that it should never have been implemented. With a more general message definition less complications would have arisen. The messages as defined for the purpose of information exchange between the software and hardware parts of the system description were sufficient, but when additional features were needed, as in this case extended possibilities to carry information, the structure was unable to meet the demands. The load regulation had to know the state of the system in detail which meant exchange of information between components that were not assumed to have any communication. The problems encountered with the implementation of the load regulation, which was not considered from the beginning of the modelling process, show that the structure, determined during the early stages of the modelling, sets a limit to what later can be added to the program.

There are in the simulation a number of independent generators, for example one for each type of call. These calls have signal sequences that uniquely determine the processing such a call requires. But there are parts of these sequences that overlap, and some signals are present in more than one of the call types. This had to be taken care of in the simulation, and it was verified that each sequence, when checked independent of the others, was followed. It could not be verified however that the simulation behaved properly when all generators were run at the same time.

The appearance of clock interrupts, job table search and execution of jobs attached to these events were also checked and verified.

Verification type II.

The simulation study had some definite goals. The performance of and load on the central processor were interesting measures. Waiting times in the central processor buffers and possible extreme values of busy periods on the lower levels should be investigated. The model should be able to predict Dial Tone Delay and similar measures involving repeated visits to the central processor and regional processors. The type II verifi-

cation was concentrated on these measures although others could have been chosen.

The expected load on the central processor's traffic handling level, i.e. the level attached to the B-buffer, could be estimated from a simple queue model. If no attention was paid to the branching of signals, and if all calls were assumed to follow the same sequence (which could be arranged by setting some random values to something extreme, either one or zero), then the number of visits to the central processor per call could be calculated. If this arrival rate is multiplied by the average signal length, i.e. the average number of instructions per signal, the average central processor load is obtained. The processor load was measured in test runs and a good agreement between estimated and measured load was obtained, which indicated a proper scheduling and job handling.

Simple performance measures for single server queues depend on the load only, and since the simulation described a queueing system that could be expected to behave "worse" than for example an M/M/1 queue, measured quantities, as for instance queue lengths and mean waiting times, ought to be larger than for this simple case. Such differences were indeed observed, but nothing could at this stage be said about the correctness of the observed values, this had to be done during the validation process.

The definition of load is in this context not obvious, but in the simulation measured as the part of the total simulation time spent on and above the level currently of interest. This measure should for JBB correspond to the probability of that queue being empty. The queue length of JBB was measured with regular intervals which were long enough for the processor to change state several times.

Dial Tone Delay consists of delays encountered in RPs and the CP. The RP times were, as already described, set to a constant for each RP group apart from a small contribution illustrating the time it takes for an RP to discover the arrived signal. The RP times are, compared with the execution times for the intermediate signals in the CP, large, and the Dial Tone Delay should consequently only consist of RP function times when the CP is lightly loaded. The Dial Tone Delay was therefore measured with a small arrival rate and compared with the expected value. The distribution of Dial Tone Delay must have a lower limit which is given by the minimum time it takes to get a call to the point where dial tone is given. The agreement of expected and measured values did also verify the measurement and the updating of necessary variables.

Waiting times were also measured and compared with queue lengths via Little's result. Some measurement errors were revealed in this process, and to compare differently measured quantities with the aid of Little's result did generally prove to be an extremely good technique.

Some minor changes were made in the program and all delays were set to be samples from some exponential distribution. The system should then act as a Markovian Queueing Network and the flow and the average service times could be verified in this way. The alterations needed were mainly the elimination of the clock interrupt and insertion of exponential distributions in CP, RP and RPH. These changes did completely ruin some properties, as for example the variances obtained

for Dial Tone Delay, but they did all the same show that queue lengths and loads were of a magnitude that could be expected.

The final type II verification that was performed is usually not possible to make, and involved comparison of results obtained from another simulation of the same system. It is rare that two comparative simulations exist concurrently and perhaps even more rare that they have been made for the same purpose and that corresponding measures could be extracted from them. This was the case however.

The other simulation was built with a quite different philosophy, and used extensive indata instead of the function blocks discussed above. This made the program smaller, but the possible checks were fewer and verification was less elaborated. In spite of this, it would be a strong argument if performance measures obtained from the two, indeed independent, simulations could be seen to agree. Test runs made under equal circumstances and for equal input were therefore organised and the results compared. Mainly mean values were compared and these were shown to have an acceptable degree of similarity. The tests were made at an early stage of the evolution of the model described here, and the comparisons were consequently not as sophisticated as they could have been to-day when an increased knowledge of the system behaviour has been gained.

The program was with these tests considered verified. The modular structure and the extensive possibilities to follow signals through the system produced great confidence in the verification process. It was assumed that no major errors still existed and this has so far (approx. two years later) not been disproved. The program has not been used to give results for all cases it is capable of and errors may hide in not yet fully used parts of the system.

5.5.2 Validation

The possibilities to validate the program in a regular way, i.e. to compare measurements performed in the simulation program with measurements conducted on the "real" system, were limited mainly by the support given in the AXE for measurements. The support can be of two types: hardware support and software support. Hardware support means that the system can be monitored without any interference with the normal functions of the system, while software support involves execution of software functions that perform the measurements. These software functions take capacity from the system and the load does consequently increase during measurement.

There is a profound difference between measurements performed on the "real" system and similar measurements performed in the simulation program. Data obtained from the real system are valuable as such, and give information about the system and how it performs under certain load conditions. It may be that this particular case is not especially characteristic for the system and that measurement under different conditions would reveal quite different behaviour, but the system has all the same performed in the manner measured, and this is valuable informati-

on. This is not true for simulation results, at least not in general. A behaviour that is observed in a simulation may not say much about the performance of the "real" system unless it is observed a substantial number of times and under different conditions.

If data obtained from measurements on the real system should be possible to use for validation purposes, they have to be analysed in the same way as data obtained from the simulation. This is sometimes not simple to accomplish. Data may not be accessible directly, but are given as computed quantities as means and variances. The thereby used statistical methods may not take sufficient precaution to handle dependent samples and other statistical difficulties. It can be extremely hard to get confident information about the way in which the measurement is performed. If the measurements are performed in software the system is investigated at a load which differs from the one emanating from the functions apart from the measurement routines. If the load created by these routines is of the same structure as the other functions of the system, then the measured values can be correct, but if it is not, little confidence can be put in the result.

Validation of the model did reveal some of these problems. The hardware support is limited to central processor load measurements and does not involve any statistical analysis. Other measurements have to be performed in software, and the load, thus created, may be of significance. Some functions are hard to control, and it is difficult to create sufficiently simple circumstances under which a validation can be made. The investigated properties are in some respects subtle and also extremely load sensitive, at least for high load which is the interesting load range.

It was mainly four quantities that were used in the validation process:

Central processor load

Waiting times in job buffers

Queue lengths in job buffers

Dial Tone Delay

The latter three are dependent on the first in that they all increase if the processor load increases. It is therefore essential to know at what load the measurement was performed. But this quantity is itself stochastic and has to be measured. One therefore ends up with a double ambiguity which further limits the possible conclusions.

Measurements were, with all the above reservations, compared and showed in general good agreement. It was mainly mean values that were compared however, and it is likely that these would show good agreement with results from simpler models as well. The validation can therefore not be said to give much information about the ability of the model to predict more subtle behaviour. But it does of course not, on the other hand, say that the model cannot give this information.

5.6 Results

This section lists some results obtained from the simulation model described in the previous sections. The simulation model was, during the active phase of the project it was a part of, used to investigate a number of properties. Some of these were investigated, however, mainly because of the efforts to develop accurate analytical formulae which could predict performance measures of interest. These attempts were rather unsuccessful in the beginning, and the simulation was used to find out where the analytic assumptions were too crude, and to estimate the accuracy of the analytical models.

But the simulation was also assumed to be able to produce results that concerned the capacity of the processing system of AXE. The capacity can be expressed by a number of performance measures, and some of these are presented below. One of the most common measures for computer systems is queue lengths in front of the processing unit. It is JBB (Job Buffer B) which receives most of the signals that emanate from the traffic handling. The mean and variance of the length of this queue are therefore of interest. From a traffic handling point of view Dial Tone Delay is an important measure and it was consequently measured in the simulation.

I/O-communication, certain standard routines and maintenance are handled by the central processor when no jobs are present at levels above JBC. These tasks will therefore not be executed if the processor stays long at levels on and above JBB. This time is equal to the busy period of JBB and higher levels and this has been measured in the program. System designers suspected that these busy periods sometimes could be longer than had been expected and were therefore interested in getting a measure of the variance of busy periods. The results are presented below and some comments follow.

The curves should not be read as representing exactly the simulation results but rather the principal behaviour. To produce smooth curves a large number of points has to be simulated, which would require unfeasible running times. The curves have therefore been estimated by an interpolation method which takes advantage of some of the extra information available for the curves. It is for example known that most of the curves start at zero (or at some lower bound) and that they approach infinity as load approaches unity. This information can be used to force the curves to stay within certain limits. The curves are also increasing with load. No confidence limits are shown since it is not the quantitative value of the points on the curve that are of interest, but rather the steepness and its relation to analytically derived measures.

Figures 5.13 and 5.14 show mean and standard deviation for JBB queue length, respectively. The mean does not show any remarkable behaviour, but the curve has a shape common to queueing systems with infinite queue. The standard deviation curve is steep when load approaches unity. This seems to be a feature common to most complex computer systems. They are, up to a certain limit, rather insensitive to changes of load. But

Fig. 5.13

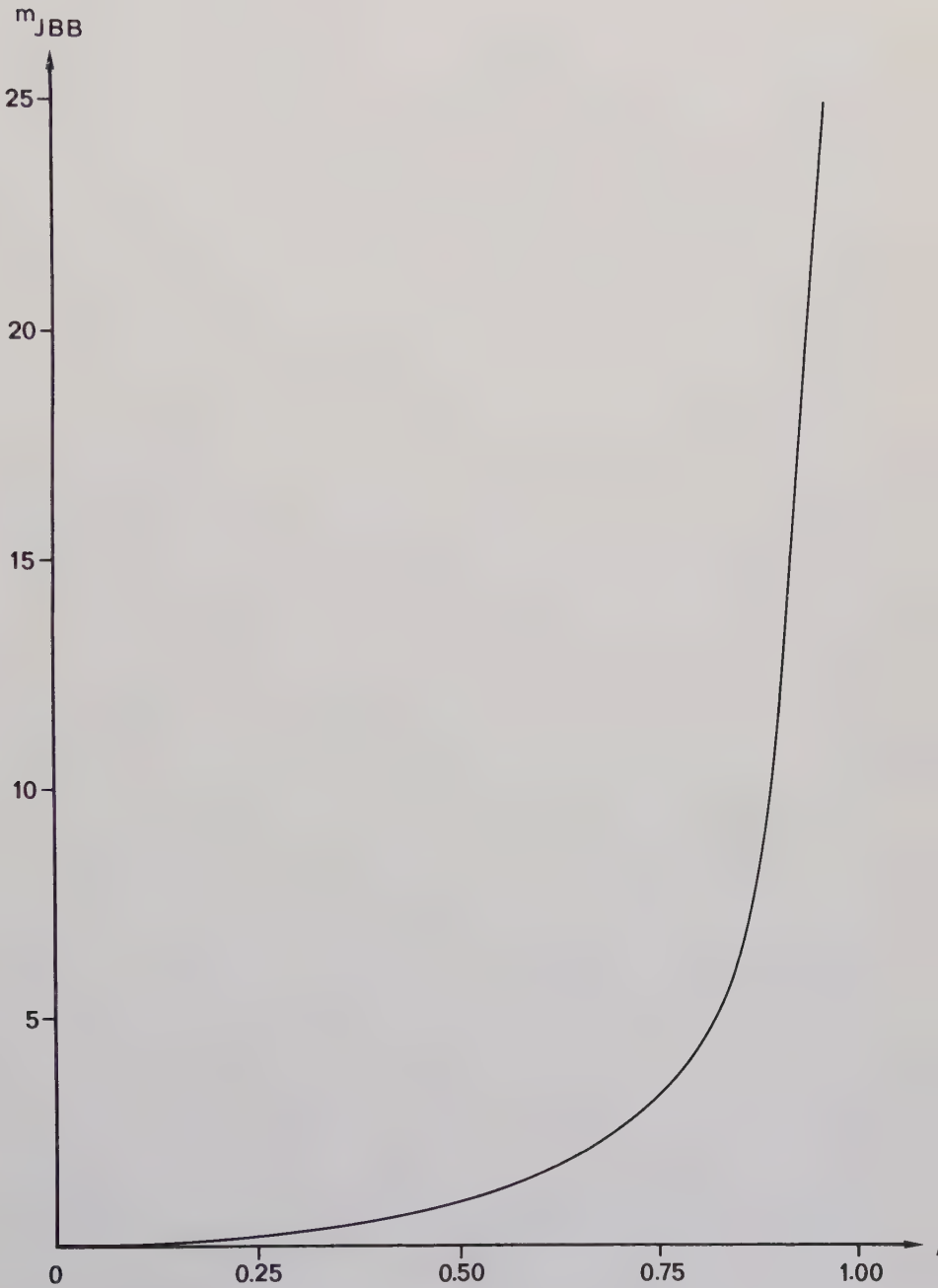


Fig. 5.13 Average queue length of JBB

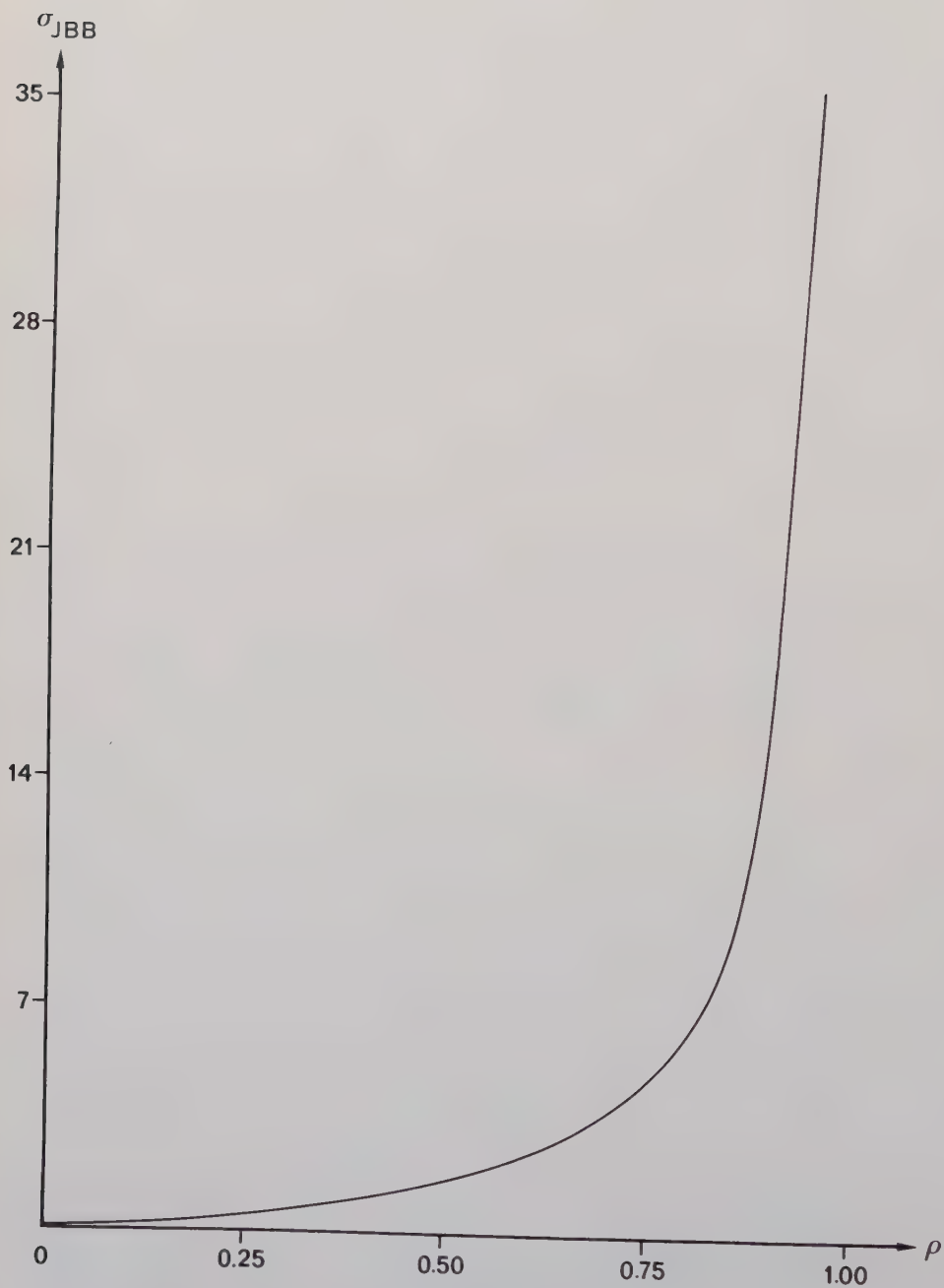


Fig. 5.14 Standard deviation of queue length of JBB

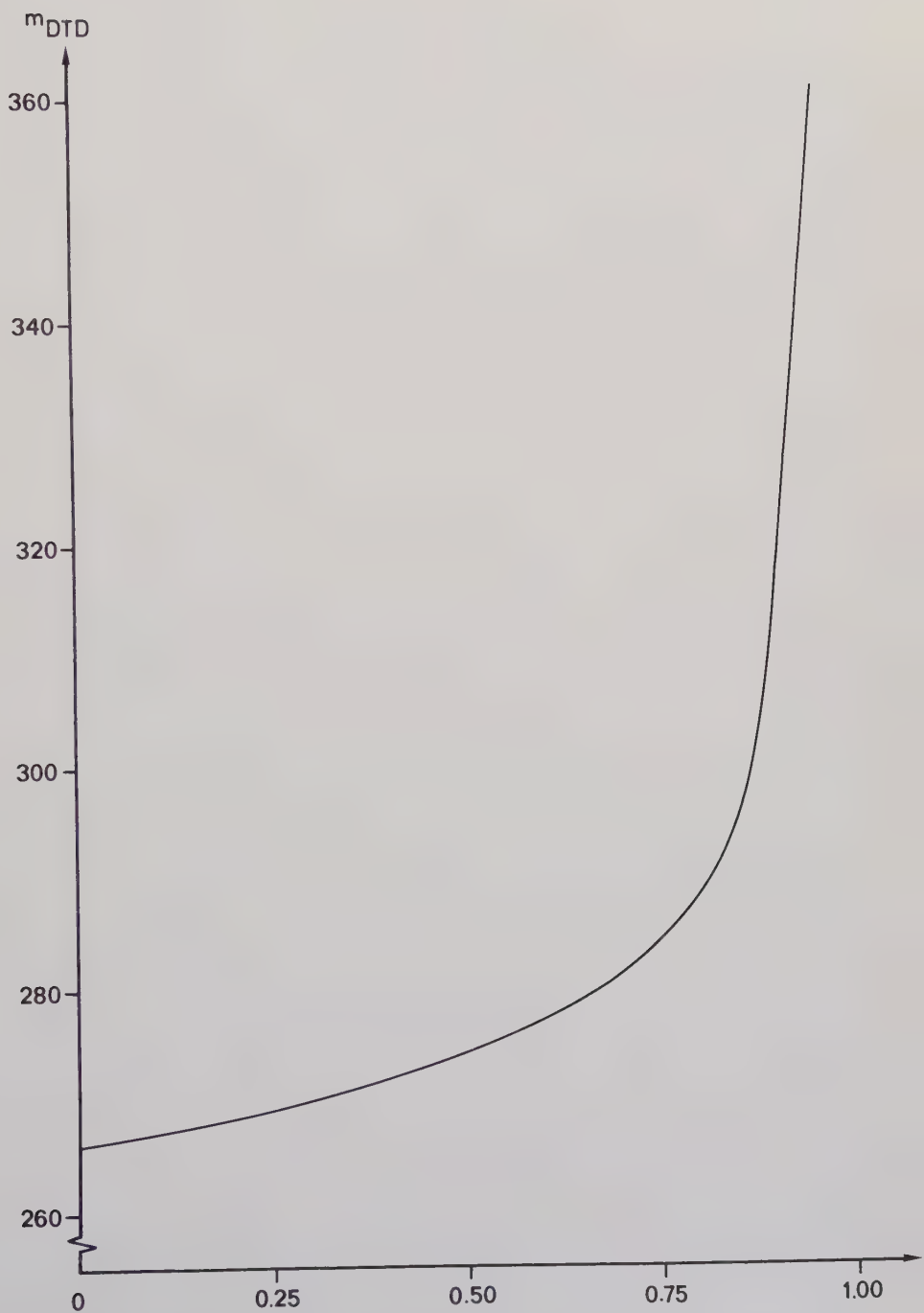


Fig. 5.15 Average Dail Tone Delay

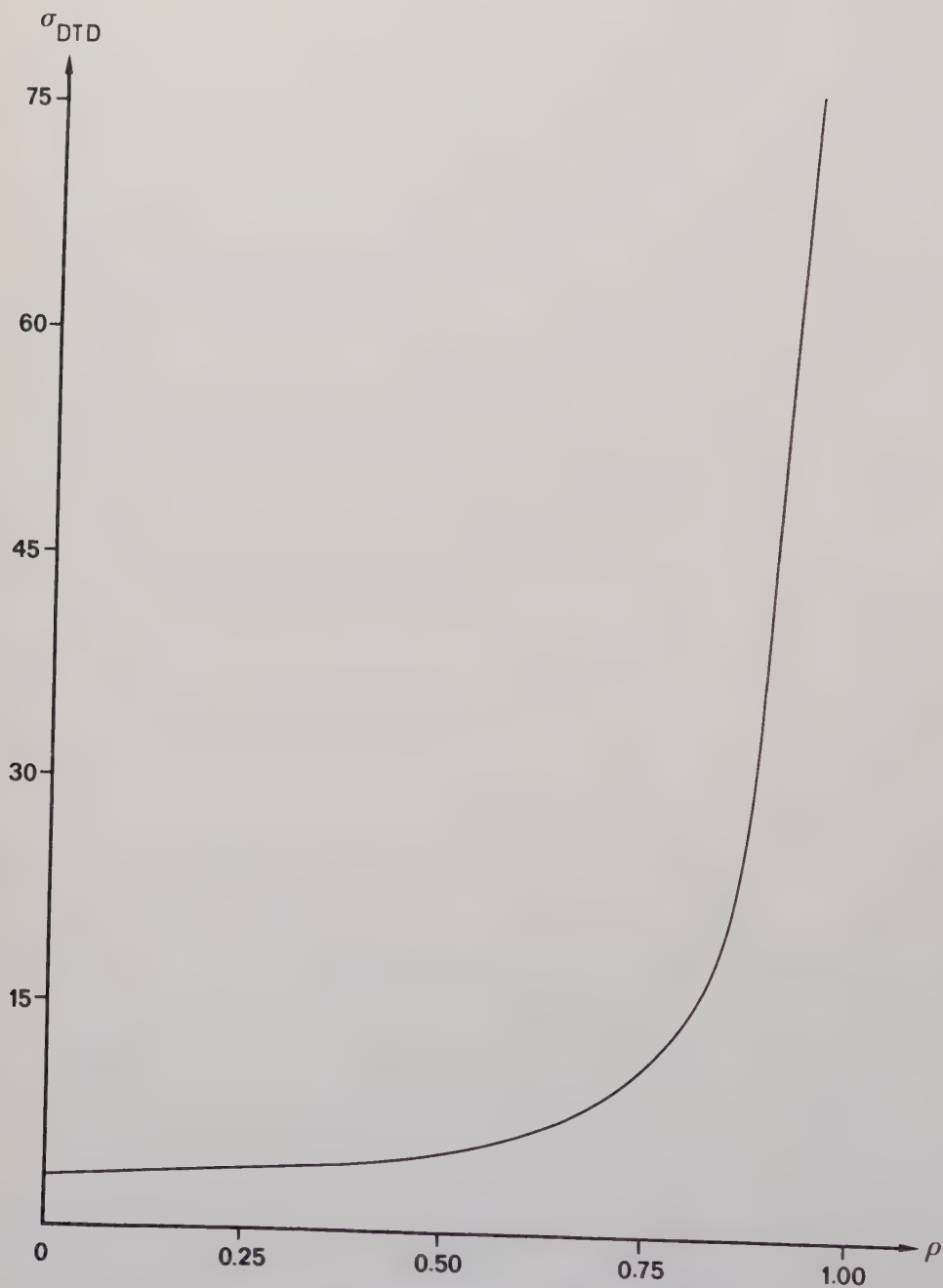


Fig. 5.16 Standard deviation of Dail Tone Delay

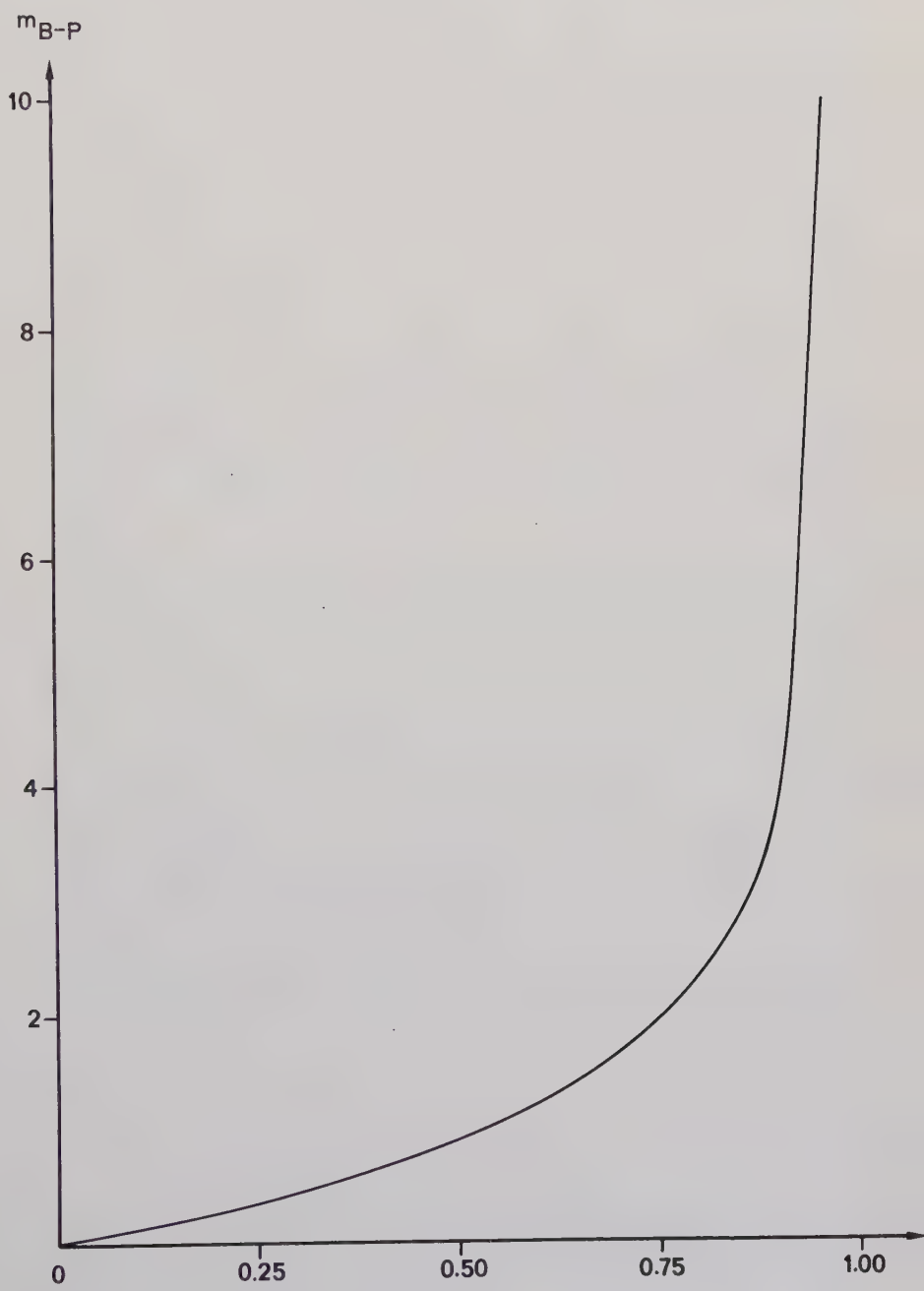


Fig. 5.17 Average length of busy periods

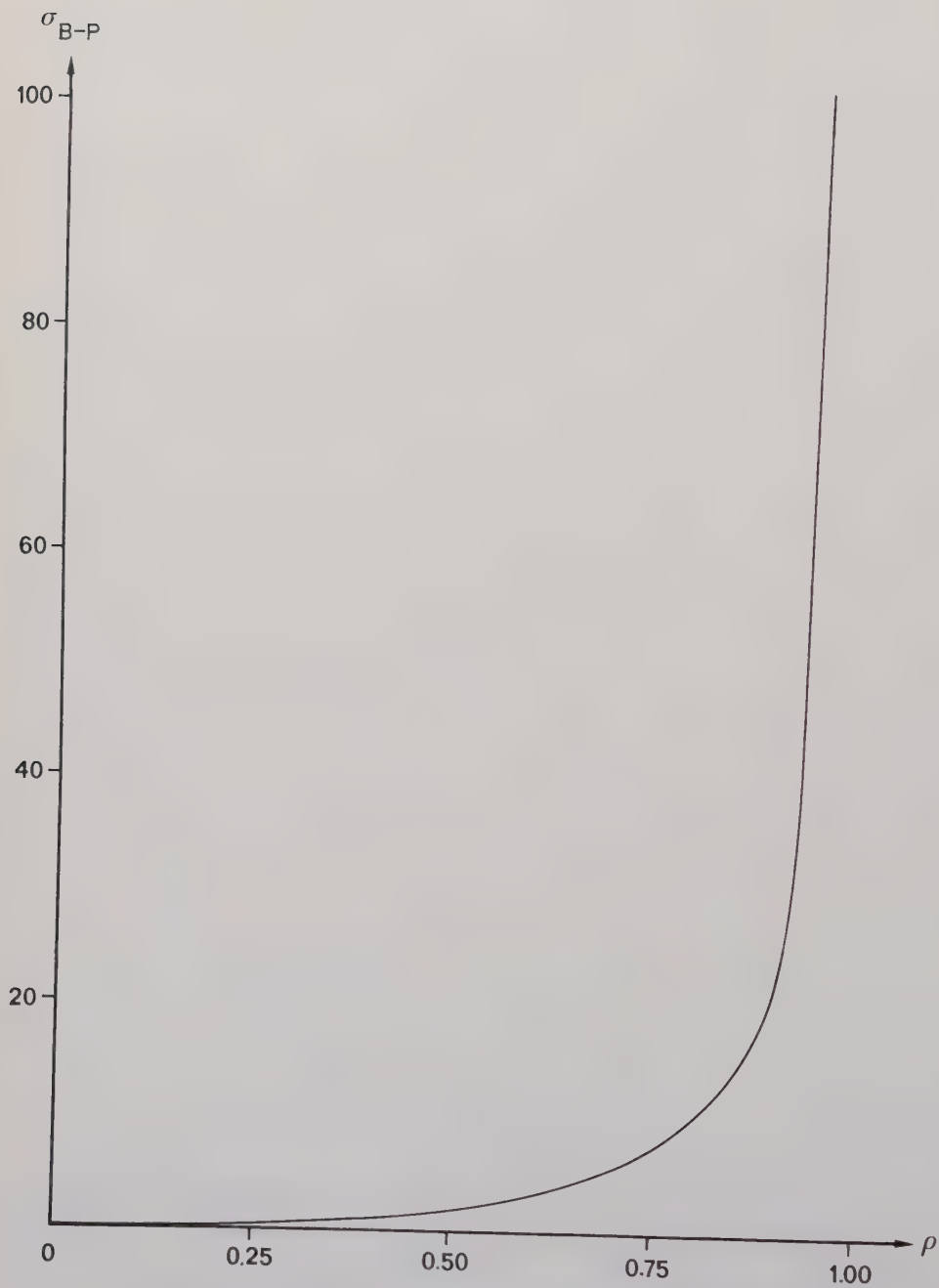


Fig. 5.18 Standard deviation of busy periods

at some point the increase accelerates, and even a minor increase of load can make the system show unfavourable behaviour what waiting times and queue lengths are concerned. The effect of a large variability of queue length is of course that the memory utilisation is small on average since the memory has to be dimensioned for the (rarely occurring) extreme cases.

With memory available at relatively low cost — which is the case to-day — large variability of memory occupation is perhaps not a major problem. A more alarming behaviour can be found in the curve showing the standard deviation for the Dial Tone Delay. This curve is also very steep in the vicinity of unity load. This means that when the exchange works at high load subscribers will be given service differently fast. Some subscribers will get Dial Tone almost directly, while others have to wait long. A particular subscriber will also, if he is often dialling at the same time of the day, have a chance of getting differently good service at different times. This is not especially good, since subscribers can very well stand a slow exchange if it is only slow all the time. They do, however, tend to get annoyed if things change from time to time.

The standard deviation curve starts at a point at which the deviation is not zero. This is due to a load independent term which emanates from the time it takes for a regional processor to discover a subscriber call attempt. The mean of the Dial Tone Delay is in the same way starting at some 270 ms which is the minimum time it takes for relays to operate and tests to be made and consequently for the subscriber to receive Dial Tone.

Figures 5.17 and 5.18 show mean and standard deviation for the busy periods of the Central processor on levels JBB and above. The curve of the mean is not alarming, but the standard deviation is extremely steep. This will have the effect that the exchange, when it works under high load, will for long periods of time be unable to execute for example standard maintenance routines. This will not happen often, since the mean of the busy periods is small, but every now and then it happens that extremely long busy periods arise. It was not, without further investigations, possible to track down the cause of this effect. The size of the main model made it almost impossible to manipulate the assumptions, the model and the processor organisation in order to find out what caused the effects. A simpler model was therefore built, but this will be presented in chapter seven, when three different models will be discussed.

The simulation model was able to reproduce data which confirmed behaviour that had been observed in measurements, for example the existence of very long busy periods.

6. A simplification scheme

6.1 General

The previous discussion on credibility brought up the question of how we, when we build and use a simulation model, gain confidence in the model. It was stated that the confidence could only be inductive in nature and that anomalies and exceptions could always be found. But it was also stated that the construction of a model was not a random process and that we already when we first came in contact with the target system considered the relevance of different pieces of information. This chapter is an attempt to investigate the relevance of different system components for the properties of interest and to form some guidelines for the modelling process.

When we embarke on a simulation task, we have to put and answer a few questions before we proceed:

- a) What kind of information do we expect the simulation to provide us with. Information specified as times, amounts and so forth.
- b) Can the simulation model we have in mind provide us with the information we are interested in.
- c) Is it only a simulation model that can give us the information, or can it perhaps be supplied by simpler and/or more reliable and efficient models.
- d) If we embarke on the task, is it likely that results will be due on time and will we have enough money and time to produce them.

These questions are related to the accuracy and complexity of the model. An extremely complex and detailed model will (if it is correct) probably give us the results we are looking for, but will also take substantial effort to develop and run, and will perhaps not produce results fast enough. A simpler model will take less time to develop, but will on the other hand not be as reliable as the more complex one, and may not be able to produce the desired information. The most adequate model does therefore seem to lie somewhere in-between, the question only being where.

The main argument for credibility of models is that they are detailed. Many reports on projects in which simulation models have been used to verify a simpler, usually analytical, model have a statement beginning: "A detailed simulation model was built ...". The authors do by this mean to say that the simulation model is more accurate and credible than the analytical model and that it thus can be used to check the validity of this. This is of course in most cases quite true, but it is also likely that the simulation model is too detailed in some respects while it is insufficiently accurate in others. A much better argument for credibility

would be: "The model comprises the properties of the 'real' system that are of importance in the considered context and which have impact on the investigated properties".

The conclusions are: that the model should be simple enough to be realisable in the allotted period of time and with the equipment and resources at hand; that it should comprise necessary behaviour but not be unnecessary complex; that it should contain behaviour that is not easily described by other modelling techniques; and that it should be able to produce desired results.

To be able to accomplish these goals, some kind of scheme has to be followed in the design process of the simulation experiment. The estimation of whether the goals can be met in the required time period and the applicability of other modelling techniques lies outside the purpose of this text. The results obtainable from a simulation study have been discussed in previous chapters, and the remaining discussion deals with the complexity of different simulation models and the relative importance of different system components and properties.

The rest of this chapter will be devoted to a discussion on the possibilities to form some kind of base for judging the importance of a specific property or component.

The discussion will focus on three important modelling issues: model balance, information acquisition and pre-modelling analysis.

6.2 Model balance

The most relevant parameter in a simulation is time. Time is the concept around which the modelling turns, and which can be more or less accurately described. A discrete event simulation centers, as the name indicates, on events, and these are measures of time. The time can be moved forward by many events occurring with small intervals or by few occurring with longer. The accuracy of a model is, generally speaking, related to the number of events used to describe an activity in the "real" system. An activity can either be described as a series of events occurring with smaller intervals (which perhaps are more easy to estimate the length of) or by one large that is drawn from a distribution illustrating the smaller subintervals.

To apply a process approach to some system means to divide the system into a number of processes. A process is a structure which "forever" goes on in a loop and which, within this loop, has a number of well-defined states in which it stays some time before it goes on to the next. The time a process stays in a state is related to the time it takes for the "real" system to perform the tasks illustrated by that particular state. Depending on how states are identified and concatenated, differently many states, and relating delays, are described. A complex and multifaceted process can contain a large number of states while a more simple usually contains less. The individual processes of a process simulation are usually

modelled and programmed almost independently. The interval between consecutive events in different processes can therefore differ greatly, and some processes may loop several times before even a single event occurs in some other process.

The modelling of a process is, as all modelling, dependent on the knowledge we have about the behaviour of the components comprising the process, and on our predetermined opinion about how detailed that particular process needs to be. The greater knowledge the more inclined we become to model a process in detail. This will cause the process to have many states and time to be stepped forward relatively slow.

All properties of interest in queueing structures depend among other things on how the time delays are modelled. From a measurement and accuracy point of view, there is no reason to model and describe some processes well, if some are based on rather crude assumptions. The point of a balanced model is to, if possible, assert all involved processes approximately the same time scale and to model them in accordance with the desired measurements. To make this point a bit more clear, measurements have to be related to how detailed the processes are described. This will be done below, and a discussion on the use of stochastic distributions will also be made.

In a discrete event simulation, time is stepped forward from one point to the next, with an amount equal to the distance in time between two successive events as they occur if the system is viewed from above, where all processes are visible at the same time. Events are therefore related to different processes and their frequency depend on the intervals with which the individual processes create new events. Quantities of interest in the simulation do often relate to some of the events of the system, and are therefore also related to the events that occur in the processes that comprise the system. The times between these events are, in the kind of simulations discussed here, random variables.

Different processes can therefore be said to reside on different levels of the system. At the lowest level fast processes, i.e. processes which have short intervals between successive events, rotate. From this level and upwards, processes have longer and longer intervals between events, and there is some process of the system that is the slowest, and which will consequently give rise to the least number of events in a simulation of given length. The time it takes to run a simulation is mainly dependent on the number of events that occur during the run. Many processes at low levels will require a large number of events and the simulation will be very heavy to run.

If now the interesting properties of a simulation of the above kind are related to events occurring at a high level, measurements will be difficult to make. The studied quantities are observed by sampling a stochastic process. It is likely that the samples are correlated, and we do therefore need a "large" number of samples to be able to make a statistical analysis of the result, and do consequently have to make a large number of measurements. Every event, which is of the type that makes it possible for us to perform a measurement, will involve a large number of events at lower levels and will therefore occur "seldom" in the simulation. If the level of detail of the processes at the lower levels could be reduced, less

time would have to be spent before the required number of events at the higher level had occurred and the simulation would be less expensive to run. The question is when this reduction can be performed without lessening the accuracy of the simulation. This will be addressed in 6.4, but before that some other problems related to model balance.

Random numbers drawn from a distribution are in simulations generated by pseudo-random-number generators. These generators do, depending on the initial value, produce "random" numbers that are predetermined, but which fulfill certain requirements what randomness is concerned. It is of course always possible to discuss when a stream of numbers should be considered random, but it is assumed that the randomness of the numbers is sufficiently good compared with other sources of error. The random numbers are used as delays in the simulation and can thereby be used at either a low level or at a high level in the hierarchy described above. If for instance a generator is placed at too high a level, few events related to that generator will occur between measurements and the result will be sensitive to the start value of the random number generator. It is therefore advisable to put the random number generators as far down as possible or to make several runs with different start values, so called replication runs. See also chapter eight.

The selection of places to put the random number generators at, is related to the second problem of unbalanced models. Random numbers are usually used to estimate delays that are hard to model in more detail or which are complicated to predict. They do therefore in a double sense represent an ambiguity, and determine in a way the accuracy of the entire simulation. If such ambiguous delays are present at one level of a simulation, it does not make much sense to model the lower levels extremely detailed unless the purpose of the simulation is to study the behaviour at some specific low level. This means that the simulation should be made as balanced as possible considering the information available and the results required. The time scales of individual processes should if possible be made to agree and one process should not be modelled without being related to the other processes in the system and to the intended measurements. If it is impossible to estimate the likely delays at a lower level, a two-level simulation is to recommend in which one simulation is used to estimate the low-level delays and another is used for measurements. This kind of simulation cannot trace individual customers, and may consequently destroy important dependences.

There is a third reason to keep the model balanced. The variable in a simulation that keeps record of the system time, i.e. the internal simulated time, is usually a single precision real variable which in many cases is not accessible to the user. This is true for SIMULA and GPSS for example. Simulations written in a general purpose language do of course have access to the time variable, but this kind of simulations is becoming less frequent and for this and other reasons the arguments below are still valid.

The time variable has, in the beginning of a simulation run, a precision which depends on the hardware of the computer. This precision is maintained throughout the simulation, but the magnitude of the variable increases constantly as the simulation runs on. If the variable at some fix time has a certain value and an event occurs, time has to be stepped for-

ward with an amount equal to the difference between time now and time at which the event should occur. The system time is updated with this difference so that system time is assigned the sum of time now and the difference. If the difference is small compared with the magnitude of the system time, truncation will take place. This means that the new system time will differ from the expected, and measurements performed in the simulation will be erroneous. The effect is of course not binary in the respect described above. The error exists throughout the simulation, but does not represent a problem as long as the maintained accuracy lies within the precision of the measurements. Similar problems arise when differences in time between events are to be measured. If a variable is set when an event occurs, arrival of a customer for example, and if that value is subtracted from the system time at a later event, exit of the same customer for example, the result may be in error if the system time is large.

These problems can of course be overcome if certain measures are taken. The system time variable can be declared in double precision. This will make the updating of the system time less critical and the time at which the problems related above arises will be put further away. Measurements of time differences will still be errorprone however, unless the variable that is set at the occurrence of an event is declared in double precision too.

Another measure, that is possible only if the system time variable is accessible to the user, is to translate all interesting variables with regular intervals. It can, if the measurement routines are properly designed, be sufficient to translate the system time, but care has to be taken to check the results. The method requires that the absolute system time is of no importance, or, if this is the case, that the number of translations is remembered so that the absolute system time can be calculated.

All attempts to overcome these problems will have negative effects on the program structure and on running times. The program will become more error prone and more complicated to validate. The process may be partly correct and minor errors, that are hard to discover, may hide in the measurements. Some variables may be "forgotten" in the double precision declaration and the possibilities to move, enlarge, and maintain the program will decrease.

Simulation programs, written in a language in which the time variable is not accessible to the user, do therefore have an upper limit for how long the program can be run before the effects of the decreasing precision become significant. A program, that has the smallest time change set to one (which always can be assumed since the absolute time only is fiction in the simulation program), will reach the upper limit after approximately 10 million events on a computer with seven decimal digits. A rescheduling of a system component will after this time result in an event at the same system time as the present, and the system time variable will not be increased. But before this occurs, accuracy of the measurements has decreased, and the optimal running time is less than 10 million events. There is a tradeoff between the number of samples that can be used to estimate a quantity in the simulation and the accuracy of the samples. The relation is not simple however, since the fastest process may have more or less impact on the measured quantity. The ef-

fects have to be estimated in each particular case, but it is likely that the simulation can be run close to the upper limit in most cases.

6.3 Deterministic analysis

Most of us do now and then wonder about the world we live in. It is at the same time ordered and random. Man has, since he first began to think and express himself, tried to find the cause of things, how they depend on each other and why they happen. But it is all so complex. Philosophy and science have not been able to give reasonable explanations of more than a fraction of life, and even these explanations are often changed or revised. When we are told that life emerged out of random meetings of molecules over billions of years, we find this hard to believe, and even if we do, we have to ask ourselves where the molecules came from and why they met in just the way essential to creation of life. Some people resign and say that this is not for man to understand, some put their faith in science and believe it is only a question of time before we know all we need to explain how life emerged, and others make up a completely new system and call it God.

Whichever position one takes, things appear to happen at random. Out of many possible events one is selected (by someone?) and it is impossible for us to tell in advance which will happen. But since we usually try to find a cause of everything, we often try to explain why that particular event was selected. This reasoning does usually lead us to some new events, which in their turn appear to be happening at random. If we continue this process long enough we do sooner or later end at events that we not possibly can see the cause of. They do either not lend themselves to be investigated or they do require more information than we can store and/or collect. Examples of the former are processes that depend on the behaviour of matter, where we cannot gain more knowledge at the moment. Why do for example cancer cells suddenly begin to grow uncontrollable and why do the volcanos suddenly erupt. Examples of the latter are the joint effect of many individual actions, as for example the occurrence one weekend of twice as many car accidents as the weekend before or breakdown of a telephone exchange because more people than usual tried to telephone. Things like these can only be observed, not satisfactorily explained. And usually, we want to do more than that. We want to predict a behaviour.

Random events can more or less adequately be expressed by stochastic distributions and processes. A stochastic process do not behave exactly as the sequence of events it tries to describe, but can reproduce certain measures and does therefore look like the original process. A simulation is a type of stochastic process that uses a sometimes rather complex set of distributions from which the distances in time between different events are selected. These distributions have to be estimated and expressed either as continuous mathematical functions or as stepwise constant probability distribution functions (PDF). If time delays belong to a limited set of constant times, then the PDF can be exactly described

as a vector consisting of the relative frequency with which the different times should be selected. All other cases must be approximated, either by a continuous function or by a limited set from which the times are selected. This approximation is, at least when the simulation is fairly detailed, possible to get sufficiently good.

But the distributions have to be estimated, and two different cases can thereby be observed:

- a) The first case is when "reality" is not known in detail. (Reality can of course never be known in detail, but the level referred to here is where reality does not appear to behave randomly). Some part of the simulation does in this case illustrate actions that cannot or should not be a part of the simulation in the respect that they could or should be modelled. Typical examples can be found in actions whose length depend on human behaviour. For how long people talk to each other in telephone, how long it takes for someone to answer a question, etc. Times like these can in some cases be broken down into smaller pieces, but at the bottom we always find times that are extremely hard to estimate. We do, in cases like these, have to rely either on our own ability to estimate the times and how they vary, or on measurements that have been performed. Whichever we find feasible, we have to fit some distribution to either our estimates or to the results of measurements. The latter process can be done in more or less elaborated ways. To fit a distribution to our own estimates is by no means simple, and does at any rate contain a great deal of arbitrariness. This has to be remembered when the rest of the system is modelled, since the ambiguity of one part is not compensated by the accuracy of another.
- b) The second type is when we indeed have possibilities to examine the behaviour of a system and to break down complex series of events until deterministic times remain. This does not imply that we have perfect knowledge about the internal behaviour of some subsystem of the investigated system, only that the time it takes for it to perform its task is constant and known in advance. The level at which this happens may be very low, but it is formally possible to describe the times involved in the actions. Examples of this can be found in computer systems and in communication networks. Provided we know the length of all packets in a packet switching network, the load (described as number of instructions to be executed in a particular node) and the priority structure, we are able to simulate the transport times of a specific packet. Such a simulation requires the packet to carry large amounts of data and the operating system of the switching computers to be described in detail, which would mean an extremely complex and heavy simulation. Simplifications are therefore made, but it is essential that these are not arbitrary, but performed in consistency with other system components.

Times of the second type could be exactly reproduced in a simulation provided that the necessary information is available when the time is to be selected. This is usually not the case however, and the constant times are therefore concatenated to form a distribution from which the times

are selected. To reproduce the exact times would in most cases mean to describe the system in detail and perhaps to picture it completely.

Information acquisition

To acquire information about a system in order to make a simulation model, is sometimes much of a random process, although it should not be. It is hard to find any guidelines to follow in the process and difficult to know when to stop. What is important and what is not? The above discussion can to a certain extent be used in this process. It is not always applicable, but can anyhow serve as the main path in the information acquisition process. How to use it is described below.

The balanced model, as described in the previous section, requires different system components to have almost equal time scales. Some processes will consist of scheduling clauses of the first type above, while others will have clauses of the second type. Some processes will perhaps mix the two kinds. When information about the system is collected, it should be continued until the individual processes of the simulation have time delays that are either deterministic or of the first type described above. This means, that the behaviour of a process should be investigated and modelled until it is either impossible to further divide the delays or until the delays are deterministic. This will for some processes result in a large number of scheduling statements, but since the investigation should be performed during the modelling phase only, and no program written with that many statements, the effort is not as large as it may seem. When the final model is decided, states are concatenated and new distributions formed out of the times present in the detailed model. The number of scheduling statements actually used compared with the ultimate number can form a measure of the accuracy of the description of a process. For a more detailed discussion on this, see (EKL 81b). To force oneself and the model down to a level at which no further modelling is possible, is a good way to acquire enough information to be able to judge the accuracy and appropriate level of each process. It can sometimes make evident behaviour that otherwise would have been overlooked or considered less important. This type of investigation is from now on called Deterministic Analysis.

The Deterministic Analysis has yet another advantage, which will be evident when the Operational Analysis will be described in the next section. It can be used to estimate the processing times of different actions in a queueing structure, and thereby to form the input of the Operational Analysis.

6.4 Operational Analysis

It is not unusual that people, who have conducted a simulation study, afterwards say that they at that time know how the study should have been performed. They have, quite naturally, gained insight in the system behaviour and can therefore be more precise when they judge the

importance of different system components and rule out behaviour as unimportant what the studied properties are concerned. If the system had been better known in advance, much effort could have been saved and simpler and more dedicated models could have been built.

But this is in essence the problem of a simulation as well as all other models. Were we only sure about what we were looking for and to what extent we could use the results, then things would be much simpler and we would be much better modellers than we are to-day. Modelling is indeed to be looking for the unknown and to form handleable pictures of "reality" which can provide us with the information of interest.

There exists, for this purpose, a number of different techniques, and the previous chapters have in part dealt with some of them. These techniques are not equally sophisticated and accurate, and some are more easy to implement and evaluate than others. This has previously been observed and used in for example hierarchical analysis (SCHW 79), where a simple analytical model is used to give a crude value of some quantity, while a more detailed simulation model is used to give a more accurate value. This technique has advantages, since it lessens the number of points necessary to investigate with a more time-consuming simulation model. But the results of the analytical model is in this case used mainly as input to the simulation model (i.e. it gives the point around which to continue the search). An alternative use of the analytic results would be to have them as guidelines in the modelling process and to include and exclude system behaviour on basis of the analytic results.

Given the identification of different processes and the service times of these, a simple analytical analysis could be used to relate the time scales, delays and load of the individual processes to each other. Such an analysis will be shown in the next section, but before that a bit more on the requirements that have to be put on the applied analytic technique.

It is unrealistic to assume that the average simulationist would learn any more elaborate analytic techniques or that any major benefits could be gained from an early analytic approach, especially if dedicated formulae had to be developed and previously unknown techniques applied. The purpose of the analysis should of course be to simplify the model and to make it less time-consuming to run. Advanced analytic analyses are complicated, require large knowledge of the modeller and can, if not an "off the shelf" analysis can be made, take substantial time to perform. An analytic technique that could be used in the early modelling steps does therefore have to be simple and easy to apply. It should in addition fulfill yet another requirement in order not to predetermine the modelling. The arguments for this are as follows:

The commonly used analytic tools, as queueing network theory, queueing theory and the theory of point processes and Markov chains, do to a large extent work with mathematical expressions of the randomness of the systems they are applied to. They do therefore require the user to assign distributions to the arrival processes, the delays and similar things. The choice is guided by both the modellers ability to solve the model and by experience gained from similar systems and previous successful modelling. Simulation modelling is not primarily limited by the solvability of the resulting model, and an application of analytic tools in the early stages of the modelling process would be unlucky, since it could

force the modeller in the wrong direction in the subsequent modelling. An analytic technique that uses only simple, observable properties of the studied system would therefore be to prefer.

The requirements stated above are well met by the Operational Analysis, introduced by Buzen and Denning (DENN 78). The Operational Analysis is the result of a longstanding tradition within the field of queueing networks theory. Longstanding is of course a relative expression, but the theory dates back to the 1950's, and is thus as old as the computer simulation technique. Traditional queueing network theory is based on a series of assumptions (the following quotes heavily from (DENN 78)).

- *The system is modelled by a stationary stochastic process*
- *Jobs are stochastically independent*
- *Job steps from device to device follow a Markov chain*
- *The system is in stochastic equilibrium*
- *The service time requirements at each device conform to an exponential distribution*
- *The system is ergodic*

Most "real" systems do not agree with these assumptions, and it is therefore surprising that the theory shows such good results as it indeed does. Validations have shown that the models can reproduce performance measures with remarkable accuracy (SURI 80), which has been a puzzlement since real systems violate many of the assumptions listed above.

These observations led Buzen and Denning to investigate whether the performance measures, for which the Markovian queueing network theory gave such accurate results, could be derived without the use of the rather difficult and sometimes unverifiable assumptions stated above. They did for this purpose set up some requirements, or operational principles, that should hold in the model:

- *All quantities should be defined so as to be precisely measurable, and all assumptions as to be directly testable. The validity of the results should depend only on assumptions which can be tested by observing a real system for a finite period of time.*
- *The system must be flow balanced — i.e., the number of arrivals at a given device must be (almost) the same as the number of departures from that device during the observation period.*
- *The devices must be homogeneous — i.e. the routing of jobs must be independent of local queue lengths, and the mean time between service completions at a given device must not depend on the queue lengths of other devices.*

They then defined a number of quantities, called basic quantities and derived quantities, and established relationships among these. These relationships are called operational theorems and form a set of permissible manipulations on the quantities. The set is not completely known, but evolves constantly as research is continued.

Many of the derived quantities do at first seem rather obvious and one has sometimes a feeling that the Operational Analysis is complicating things that indeed are quite obvious. This might be true for someone who has been brought up in the realm of stochastic analysis, since most of the results obtainable from Operational Analysis can also be found by means of a stochastic analysis, although the notation is somewhat different. Moreover, stochastic analysis has evolved further and can, with the stronger assumptions it employs, give more elaborate results. But there is of course a danger in this. Analytic methods can, once the basic assumptions are made, give results that are accurate. Still the assumptions, on which the analysis is founded, may not be well met in the "real" world, and the precise results may therefore be of little value. The Operational Analysis is an attempt to relate the information acquisition process to the derived results. Properties that are not easily observable and testable should not be made a base for the analysis.

This is exactly what is needed for a simple pre-programming analysis of a simulation model. We need a simple theory, which gives a rough estimation of the system behaviour. It should depend on assumptions and information which can be obtained without predetermining the subsequent model. It should deal with performance measures that do not extend beyond those possible to obtain in the simulation model.

What should then this pre-programming analysis with the aid of Operational Analysis look like? Simulation is used to estimate what impact the randomness of the processes has on the performance measures. Processes in a process simulation of a queueing structure act as servers. Some servers contribute greatly to the performance measures of interest, while others act more as constant delays of shorter or longer duration. Servers that are lightly loaded have little impact on the measures and can therefore be excluded from the simulation. (This argument applies under certain circumstances, and a more thorough discussion will follow in the next section, where the Balanced model, Deterministic analysis and Operational Analysis are tied together). The Operational Analysis can be used to estimate certain simple quantities which in their turn can be used to judge the importance of a system component.

Most of the systems considered in simulations are open systems, i.e. jobs enter and leave the system and the number of jobs in the system varies during the observation period. Operational Analysis has been applied mostly to closed systems in which the number of jobs is constant. It is not generally agreed that the more sophisticated results of Operational Analysis apply to general networks, but since these results will not be used in the kind of analysis attempted here, there is no reason to discuss the validity of Operational Analysis in a larger context.

Below is a brief explanation of the parts of the Operational Analysis which will be used in the pre-programming analysis suggested in the next section.

Operational variables can be either basic quantities or derived quantities. For a simple single server queueing system there are four basic quantities:

T = the length of the observation period

A = the number of arrivals occurring during the observation period

B = the total amount of time during which the system is busy during the observation period

C = the number of completions occurring during the observation period

From these, four derived quantities arise:

$L = A/T$ = arrival rate

$X = C/T$ = output rate

$U = B/T$ = utilisation

$S = B/C$ = mean service time

It is mainly two quantities that are of interest in the present context: load or utilisation and time in system. The load is a derived quantity and has already been defined as B/T , the fraction of time the system is busy. This definition is not especially fit for our purposes however and let us therefore state an alternative expression.

Under the assumption of job balance, i.e. that the number of arrivals during the observation period equals the number of departures, we have that

$$U = L \cdot S$$

which is an operational theorem.

The operational quantities L and S can be estimated by performing an analysis as suggested in the previous sections. 6.4 will show how to use the results.

The above holds for a single device, but it is simple, by indexing the variables, to extend the relations to a network with many servers.

In such a network, jobs may enter and leave the system at any device i , i.e. as C_i/T where

$$C_i = \sum_{j=1}^K C_{ij} \quad i = 1, \dots, K$$

and

K = Number of devices

and

C_{ij} = Number of times a job requests service at device j immediately after completing a service request at device i .

It follows that the utilization law

$$U_i = X_i S_i$$

where S_i = mean service time at device i , holds at every device.

The Operational Analysis gives, in addition to the results given above, relations for a number of quantities. These are not of interest in the analysis performed here and are therefore not repeated. They can be found in (DENN 78).

It is likely that the results of the Operational Analysis have greatest validity for light load where the assumptions can be assumed to hold. The results will therefore not be used to estimate performance measures in situations with high load.

6.5 Balanced Models; Deterministic and Operational Analysis

The main problem of simulation modelling is that formally everything is possible. Analytic models are almost automatically limited by the potentials of the technique, and the risk that models become too accurate and/or too complex is small.

If it could be proven that accurate simulation models indeed were correct, and that they produced credible results, problems would be less. One would just, considering the time, money and computer resources available, make as good a model as possible.

In reality, things are more difficult however. Extremely complex models tend to provide the user with little confidence, are costly to run and difficult to extend and alter. It is necessary therefore to impose restrictions on the model. Not every detail of the studied system can be incorporated in the model; unknown behaviour has to be estimated, sometimes with a high degree of uncertainty; running times have to be limited. A question that promptly arises is how to simplify and limit the model. It is of course not a good approach to use the apparent restrictions given by analytical modelling, since an analytical model just as well could be used in such a case.

Besides their inability in some cases to yield sufficiently accurate results, analytical models have advantages and disadvantages. Analytic modelling is a well-established discipline. It is relatively well known what analytical models can and cannot give, and it is also known that new results are not easily produced. Major breakthroughs in the future will, moreover, require extensive theoretical works. The analytic technique is rather complicated to adopt, and cannot be expected to spread outside a circle of professionals.

The simulation technique on the other hand, is seemingly simple to learn and is widely used as a tool for performance analysis and prediction. The real benefits of the technique are often much less than the expectations however, and many models are never used for the purpose for which they were designed. The reason for this is probably that the expectations and the goals are put too high.

The demands that come out of the above arguments are in many ways contradictory. A combined simulation and analytic approach would be beneficial if a proper level of detail could be asserted in the simulation. An analytic approach could be used to do this, but the analytic approach is hard to learn and requires the analyst to have substantial knowledge of subjects like stochastic theory, mathematics and the like.

The Operational Analysis, briefly described in the previous section, does not suffer quite as much from these restrictions. It is rather simple to understand, and can be applied without any detailed knowledge about the above subjects. The rest of this section will be devoted to an effort to form a modelling scheme out of the thoughts presented in the previous three sections.

The first step of a simulation study consists of formulating questions to which the simulation is supposed to give answers. Let us assume that the desired information is of a kind that can be obtained from a simulation.

The second step is in that case to acquire information enough about the system to be able to build a simulation model. Since the level of detail is not pre-determined, "enough" is a relative word, and it is hard to know when to stop. This is where the Deterministic Analysis of 6.3 comes in. The information gaining process should not stop when "enough" information is collected, but be continued until one of the two levels described in 6.3 is reached. This approach has two distinctive goals: to prevent the information gaining process from being stopped too early, thereby hiding important behaviour, and to give information about the service times of different components.

The third step is to set the border between the simulation and the environment and to illustrate the behaviour of the environment by some generators as earlier discussed. The generators do often represent human behaviour or the behaviour of processes not directly involved in the studied system. It can generally be stated that the border is set at a point where the feedback is getting looser. By this is meant influences that are not easy to describe and which have no direct effect on the system behaviour.

The generators have to have some intensity with which they emit new entities into the simulated system. These entities will circle in the system for a while and die eventually. A flow is thus created in the simulation and this will also be used below to estimate the importance of different system components.

It is important to observe that the arguments put forth in this text apply to queueing structures, but not necessarily to other structures. One has to remember also that the output from the simulation should be queue lengths, waiting times and similar measures. The objective of a simulation can, as already mentioned, be something else, and in such a case do the above and following arguments not necessarily apply. It is likely, however, that many of the arguments and part of the proposed scheme can be used in other simulation studies as well.

In queueing structures, jobs are routed among a number of service centers. Each service center is in a process approach modelled as a process, which takes care of an arrived customer, queues him up and processes him when it gets his turn with the scheduling discipline used. Apart from the processes describing the service centers, there may be processes which describe for example the routing or a protocol to follow. The processes treated here are the service center processes, since we are interested in simplifying the model what running times, complexity and similar measures are concerned.

The Deterministic Analysis gives the service times of the individual service centers. The analysis brings us down to a level where we either know the exact service time or where we have to use a distribution.

The generator output and the routing scheme give us a good estimate of the arrival rate of customers at each service center. This arrival rate can of course be state or time dependent. For each state it is determined, however. (Otherwise we could not program the model).

The fourth step on the road to a complete simulation model would at this stage be to model all the servers in accordance with the information gained from the Deterministic Analysis. But before such a task is embarked on, two questions have to be put:

- a) Should all the service centres be incorporated in the simulation.
- b) Should the level of detail obtained by the Deterministic Analysis be kept, or should simplification take place.

Operational Analysis is used to answer the first question and the Balanced Model concept to answer the second.

When the servers and the environment are determined, Operational Analysis is applied, no matter what simplifications this may require. Feedback, dependent service times, blocking and similar restrictions are ignored and the servers are assumed to fulfil the assumptions of the Operational Analysis. The service times are given by the Deterministic Analysis and the flows of the routing and the Generators. The Operational Analysis can, among other things, give information about the load of the individual servers. The general rule is: omit a server if its response time is small compared with other response times of the queueing structure and if its load is small. Both these requirements have to be met for an omission to be legal. The arguments for the omission are as follows:

A lightly loaded server will in most cases only contribute to the total delay with the service time of the server. If this time is small, a minor error is made by omitting it. The server will, in the long run, not form queues and will consequently not show great variations in the response times. The distribution of the service time is not, as shown in (GORD 80), crucial, and if only the mean of the service time is estimated with reasonable accuracy, results will be accurate. This is true for light load. The estimates can at higher load be quite sensitive to parameter prediction, and an omission can therefore not be advised.

It can of course be argued that the Operational Analysis is too crude a method to be used in the way suggested here, but as R. Suri has shown (SURI 80), the technique is quite robust and can be used on systems that violate the assumptions of classical queueing theory heavily.

The scheme suggested above answers the first of the two questions previously put. The question of how well the now remaining servers should be modelled cannot be definitely answered. Some general hints can be made however. The concept of model balance tells us that the simulation has a certain maximal time span, and that there is no reason to simulate events on an extremely low level if the performance measures of interest form at a much higher level. The states of the remaining processes do therefore have to be concatenated until an acceptable level is reached.

Besides this we know from the law of large numbers that sums of a large number of stochastic variables form a normal distribution in the limit independent of the distribution of the individual variables. We also know that the number of variables do not need to be large for the approximation to be good. A problem with the normal distribution is that it extends indefinitely both upwards and downwards from the mean. This can in a simulation give rise to negative times and the normal distribution must therefore be truncated or some other distribution, for example an Erlang distribution, used.

The Operational Analysis made us ignore certain important properties of the system. The servers that remain after the Operational Analysis should of course not be modelled as simple as assumed in the Operational Analysis, but should be made as accurate as possible considering the necessary concatenation and the information available. The question of where the thrust should be put still remains however. Some very good guidance can be gained from the approximate solutions of queueing network problems. The capabilities of the analytic technique are sometimes too limited since even if the problem can be accurately defined, the chances are small that an exact solution can be found. The analyst can in that case take two actions: he can either simplify the problem until it gets tractable, or he can try to solve the original problem approximately. Both methods are being used, even if the former is more frequent.

There is a number of system properties that have gained interest in the literature and which have been approached by approximate methods. A good overview can be found in (CHAN 78), where state of the art of approximate analysis is reviewed. As the authors point out, approximate analysis, and indeed simulation, is a matter of personal preference. Not all aspects of a system can be modelled, and the pick is biased by the analyst's opinions, experiences and capabilities. A checklist will therefore contain items that can be of importance, but may very well leave out system properties that have great impact on the behaviour and on the other hand contain unnecessary items. The list should therefore be read with care, and not be considered exhaustive.

When a server is modelled, it may be important to incorporate behaviour of the following kinds:

Service time distributions and dependent service times

Order of service

Priorities

Overlapping service requests

Blocking (finite waiting room)

Scheduling

Parallelism, forking and joining

Flow control and routing

Problems related to these properties have received considerable attention in the literature and many have been solved approximately with good accuracy. It is advisable to check these methods before embarking on a simulation. A large number of references can be found in the above work.

7. An alternative approach

7.1 General

A detailed simulation model of the AXE system was described in chapter five. This description showed a rather common way to simulate a system and some results that described the system behaviour were presented. The model and the program became complex and heavy to evaluate. An average run took more than an hour to make and since several runs had to be made for each point of interest, running costs were very high.

The previous chapter, number six, presented a simplification scheme that could be used if the properties of interest were common performance measures as queue lengths, delays and utilisation. This technique will now be applied to the AXE system and a simpler model will be presented. A question of interest in this context is of course whether the simplified model can provide us with the same information as the more complex and how far the simplification can be carried before the approximations become too severe.

In order not to bore the reader with even more details than presented in chapter five, only a single traffic case will be considered. The presentation will also omit many details of the Deterministic Analysis, and it is assumed that the analysis is carried on until one of the two levels earlier described is reached.

A number of models, with increasing degree of simplification, were built. The simplifications will be explained with reference to chapter five, where a more detailed description of the system can be found.

7.2 A simplified AXE model

The main objective of the previously described AXE model was to investigate the performance of the system what traffic handling capacity, Dial Tone Delay and utilisation were concerned. The performance measures used to express these properties were the relation between Central processor load and arrival rate, mean and variance of Dial Tone Delay and waiting times and queue lengths at the Central processor. These measures should of course be the objectives of a simplified model as well, and the following presentation is therefore concentrated on Dial

Tone Delay and Central processor queue length. Other measures could have been chosen, but since the comparison of results obtained by the "large" model and by the simplified model was essential, related measures had to be chosen.

The AXE model of chapter five contained four load dependent processes: the Central processor, the Regional processor handler's input and output parts and the Regional processors. A renewed investigation of the important processes of a simulation model would yield the same result and we therefore immediately proceed to the next step of the suggested investigation: the Deterministic Analysis. Let us investigate the above components in turn.

7.2.1 Deterministic Analysis

An investigation of the Central processor will result in the execution of instructions as the "deterministic" level. An instruction takes a specific time to execute if we assume that the instruction is not interrupted by a signal at the micro program level. Micro program interrupts may occur, but let us, for simplicity, stop at the instruction level.

If all instructions are assumed to take a defined time to execute, sequences that are not interrupted will take a defined time to execute. We therefore find that the deterministic level of the Central processor is the execution of a sequence of instructions, i.e. a job. The mean length of a job in the AXE version studied here, is approximately 0.25 ms. This time is thus the time scale of the Central processor job execution. The central processor contains, as mentioned in 5.3, a number of priority levels. A time queue, the job table, is searched at a higher level than that at which traffic handling is made. An investigation similar to the above shows that the time scale of this level is almost equal to that of the lower level, and the Central processor does consequently work with this time scale when considered as a whole. The deterministic level is in this case as well execution of instructions.

The Regional processor handler works at the same frequency as the bus system. The lowest deterministic level is the level at which a signal is transported either to or from an RP. The bus operates at one MHz and a signal of average length takes about 30 micro seconds to transmit.

For the input part of the RPH a second deterministic level is at the polling level at which the polling mechanism goes from one group of RPs to the next. Such a step takes about two micro seconds per RP, and a turn is for a large exchange completed in about one ms. A signal that arrives at an empty system will therefore have to wait on average half this time to receive service. The RPH can get in conflict with itself and does sometimes have to wait for one task to be completed before it can turn to the next. This can sometimes cause the data flow to be interrupted, but the wait is small compared with the other times given above.

The output part of the RPH works as an ordinary single server. It fetches signals from a buffer in the CP and transmits them when the bus gets free. The deterministic level is transmission of a signal, and the time scale will be about 60 micro seconds.

The Regional processors, the RPs, are processors that work in a fashion similar to that of the CP. The average length of a job is for the RPs approximately 100 micro seconds. Such a job may at several instances be interrupted by wait for some hardware, like a relay, to operate. This wait is load independent however and the number of executing jobs will on average be the same as if the wait did not exist. The waiting times may be of importance in the final model, but can be omitted in the Deterministic Analysis since they are not load dependent. Execution of jobs can be viewed as a deterministic level and the time scale of the RPs is approximately 100 micro seconds.

The generators

The outside world, which creates the flow of the simulation, is in the AXE case illustrated mainly by the arrival generators. The border between the simulation and the outside world is not arbitrary, but is placed at an, in this case, obvious point. A border placed further away would have the subscriber behaviour incorporated in the model, which is quite unfeasible, and a closer model would leave out some of the hardware, which would limit the applicability of the model.

The AXE system is supposed to work in an environment consisting of several thousands of subscribers. To illustrate the arrivals at the system with a Poisson stream does therefore seem to be a reasonable model. Such a model can be extended to illustrate several different groups of subscribers.

The characteristics of telephone traffic will give rise to an arrival rate of approximately 50 calls/second for a large exchange, which means that an arrival generator has to emit a new arrival every 20 ms. The assumption of Poisson arrivals implies the interarrival times to be exponentially distributed, and the time scale of the arrival generator is in the range of 20 ms.

The simulation model must also comprise generators that illustrate sending of digits, charging and similar activities, but these do all work with time scales that are much larger than the scale of the arrival generator. The discussion about these are therefore omitted, but a similar analysis can obviously be performed. The arrival generator is the main source of flow in the system. Generators which send digits do only continue the flow created by the arrival generator, and do consequently not add to the total flow. The description in chapter five, of the establishment of a call, showed that a sequence of signals was sent between different function blocks. Establishment of a call demands about 30 signals to be sent. A call can therefore, on average, be said to give rise to 30 signals, even if these are not all sent when the call arrives, but sequentially as the call propagates through the system. If the four processor parts of the system are illustrated separately, and connected to show the flow of signals, a figure as the following is received:

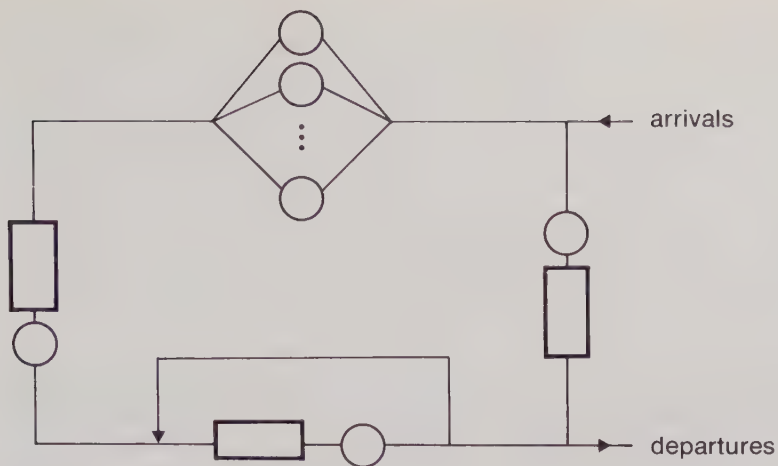


Fig. 7.1 Simplified AXE model

Calls arrive at an RP and are routed in the system until they finally leave. If an arrival rate of 50 calls/second is assumed, $30 \times 50 = 1500$ signals leave the RPs for the CP every second. All of this flow is routed through the input and output parts of the RPH, while it is divided among many RPs. An exchange that is capable of handling a traffic of 50 calls/second is equipped with approximately 50 RPs. Since the total flow can be assumed to be distributed equally among the RPs, each RP will receive some 30 signals per second, and some 50 jobs will be executed in every RP per second.

The Central processor contains a feedback loop which will increase the average service time. On average, a job is routed back to the tail of the queue 1.7 times, and the average service time will be $1.7 \times 0.25 = 0.43$ ms.

7.2.2 Operational Analysis

When the approximate service times and the flow created by the generators are estimated, it becomes possible to use the subset of the Operational Analysis described in the previous chapter to get a picture of the load on each of the four components.

The Central processor receives at a call rate of 50 calls/second 1500 signals/second. The processor will therefore execute 1500 jobs/second, each having an average service time of 0.43 ms. $1500 \times 0.43 = 795$ and the Central Processor will consequently be busy approximately 795 ms/second. The Central processor load is therefore approximately 80 percent, which is a high load. The Central processor can obviously not be omitted from the simulation model.

The input and output parts of the RPH do also have to handle 1500 signals per second. If the longer service time of the output part is assumed for both parts, then the average load will be 60 micro seconds times 1500 per second, which is about 9 percent. This is a very light load and the RPH could therefore be considered for omission.

1500 signals divided among 50 RPs will give each RP an arrival rate of 30 signals/second. An average RP job has a service time of 100 micro seconds and the RPs will thus have an extremely light load. ($30 \times 100 = 3000$ micro seconds/second = 0.3 percent). The RPs have, in the same way as the Central processor, a lower limit for the load since it always works with standard tasks as checking subscriber points and similar duties. This load will almost entirely determine the load of the RPs. This load is more or less independent of the arrival rate however, and will act merely as to decrease the capacity of the RPs with some fix amount. The RPs can consequently be assumed to have small impact on the performance measures of interest.

The final simulation model must comprise the function times of the RPs however, but this can be accomplished without any RP processes. The RPs can moreover not notice the arrival of a signal but at the beginning of a clock interval. This will cause the signals to suffer a delay that is equally distributed over the clock interval. Such a delay is load independent as well, and can be added to the function times.

7.2.3 Model balance

A look at the model balance will reveal the polling mechanism of the input part of the RPH as the fastest process. It works at a level of some micro seconds.

The other processes range upwards from around 100 micro seconds, and this seems to be the time scale at which the simulation has to be run. The section that follows will show the implementation of some different models that have an increasing degree of simplification. The models will have some of the processes omitted and the time scales will be compressed.

A look at the checklist at the end of chapter six will indicate that the models have some of these mechanisms. The AXE system has indeed dependent service times, a pre-determined order of service, priorities, parallelism and flow control.

Simplified models can drop one or more of these properties, but since it cannot be known in advance what importance they may have, omissions have to be made with care.

7.3 Models and results

This section will list some results obtained from models with different degrees of simplification. As simplification is increased, validity is in some respects decreased. It is evident that a model, which omits some parts of the target system, cannot reproduce and predict data for those specific parts. The models presented here are consequently not structurally valid in the respect of Zeigler. They do also have a smaller ex-

perimental frame. But to be structurally valid is not the aim of the models. Their aim is to reproduce specific performance measures.

The first simplified model maintains the time scale of the "large" model presented in chapter five, but omits the input and output parts of the RPH. It also omits the RP processes, but since this was done already in the original model no data exists that could be used to estimate the significance of this simplification. A model which would incorporate the RPs as well would be many times as complicated as the "large" model and even more difficult to handle and time-consuming to run. Such a model has therefore not been developed, but the conclusions have to be made on the other simplifications.

The simplified model keeps the trace of a call, but stores it in a more coded form. This made the program faster to run, but also more difficult to read and change. The program is in all but the omission of the RPH processes a true copy of the larger model.

This simplified model was developed at the research institute of the Norwegian Telecommunication Administration. A description can be found in (RYGH 82). This model is called alternative model 1.

The second alternative model (called alternative model 2) was developed at the same department as the original model (the original, "large", model is hereafter called the conventional model). In this, a number of simplifications were made. The original purpose of this model was to use it for investigation of the relevant properties of the extreme behaviours found in the conventional model. It was later used also for comparing different degrees of simplification when the simplification scheme of the previous chapter was followed. The results presented below refer to both these investigations.

The second alternative model omits the exact signal sequences, since the Central processor, after the omission of the RPH, had become the fastest process. The exact sequence was substituted by a distribution, and an exponential distribution was used, not out of convention, but because the first three moments of the distribution produced by the signals as a call progresses in the system, agreed fairly well with an exponential distribution. Some separate digit generators were also omitted and the digits were assumed to arrive in a row in the respect that the second digit arrived when the first had been processed and so on.

Figures 7.2 through 7.4 show the mean and standard deviation for JBB queue length and standard deviation for Dial Tone Delay. Busy periods were not measured in the first alternative model and comparisons can consequently not be made for this measure.

The models compare surprisingly well for all performance measures. There is a significant difference, but it is not as large as could be expected.

An even more surprising effect is perhaps that the curves do not appear in the same order as perhaps could be expected from the simplifications. Since the conventional model for all investigated cases has produced data which have been larger in magnitude than those obtained from simple analytic models, it would be reasonable to expect data from simplified models to go in the same direction. This is not the case however.

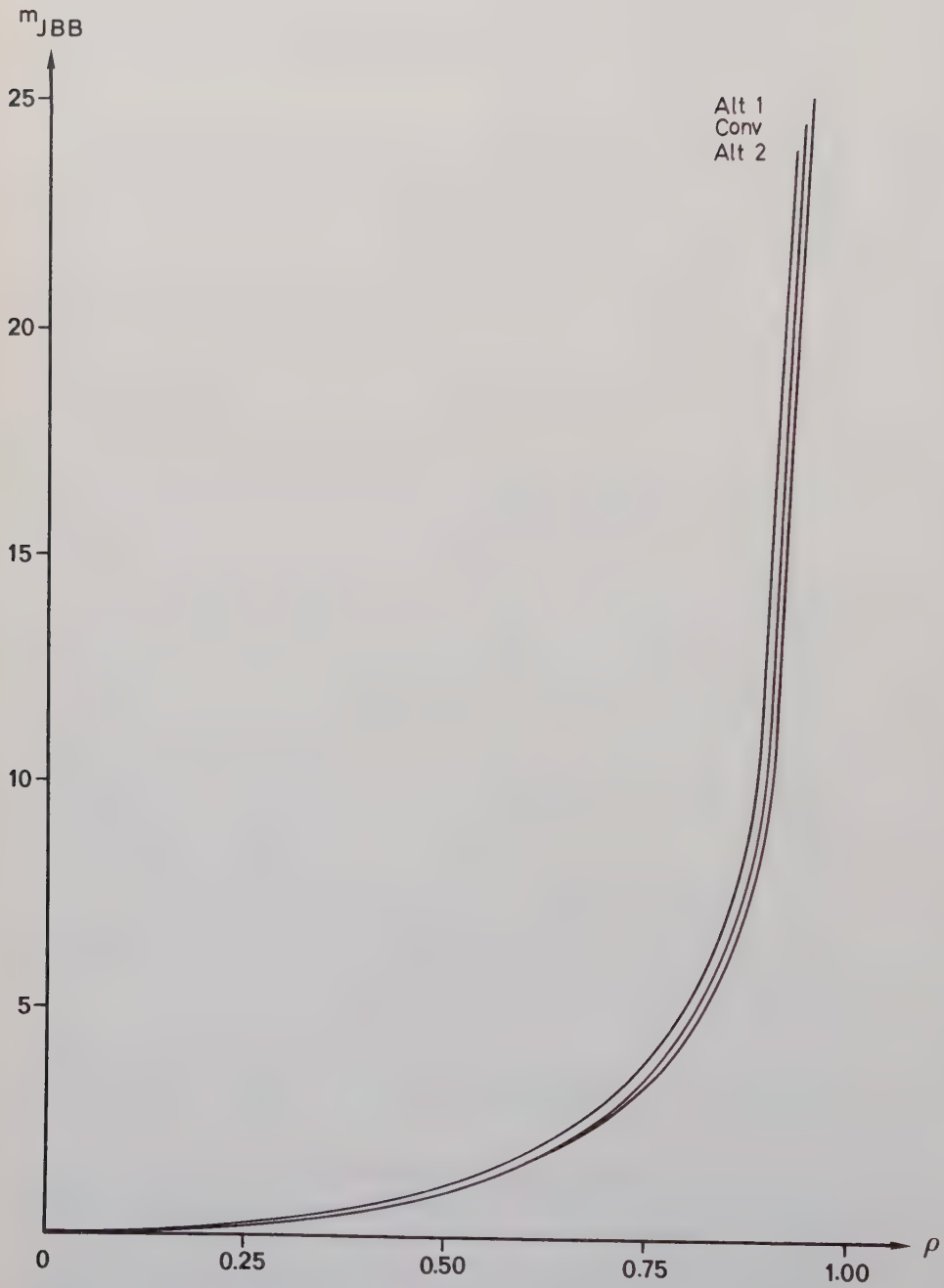


Fig. 7.2 Average queue length of JBB

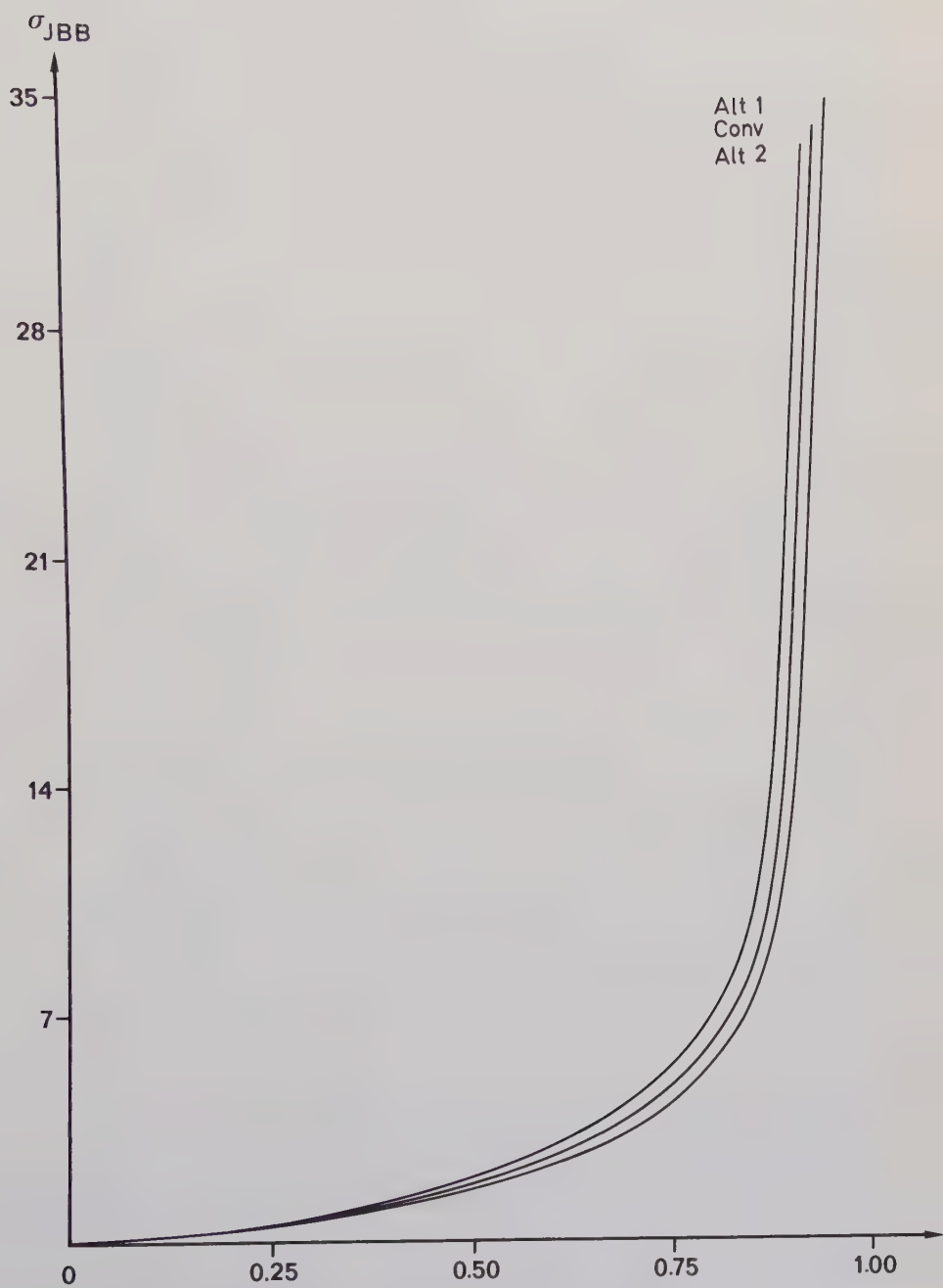


Fig. 7.3 Standard deviation of queue length of JBB

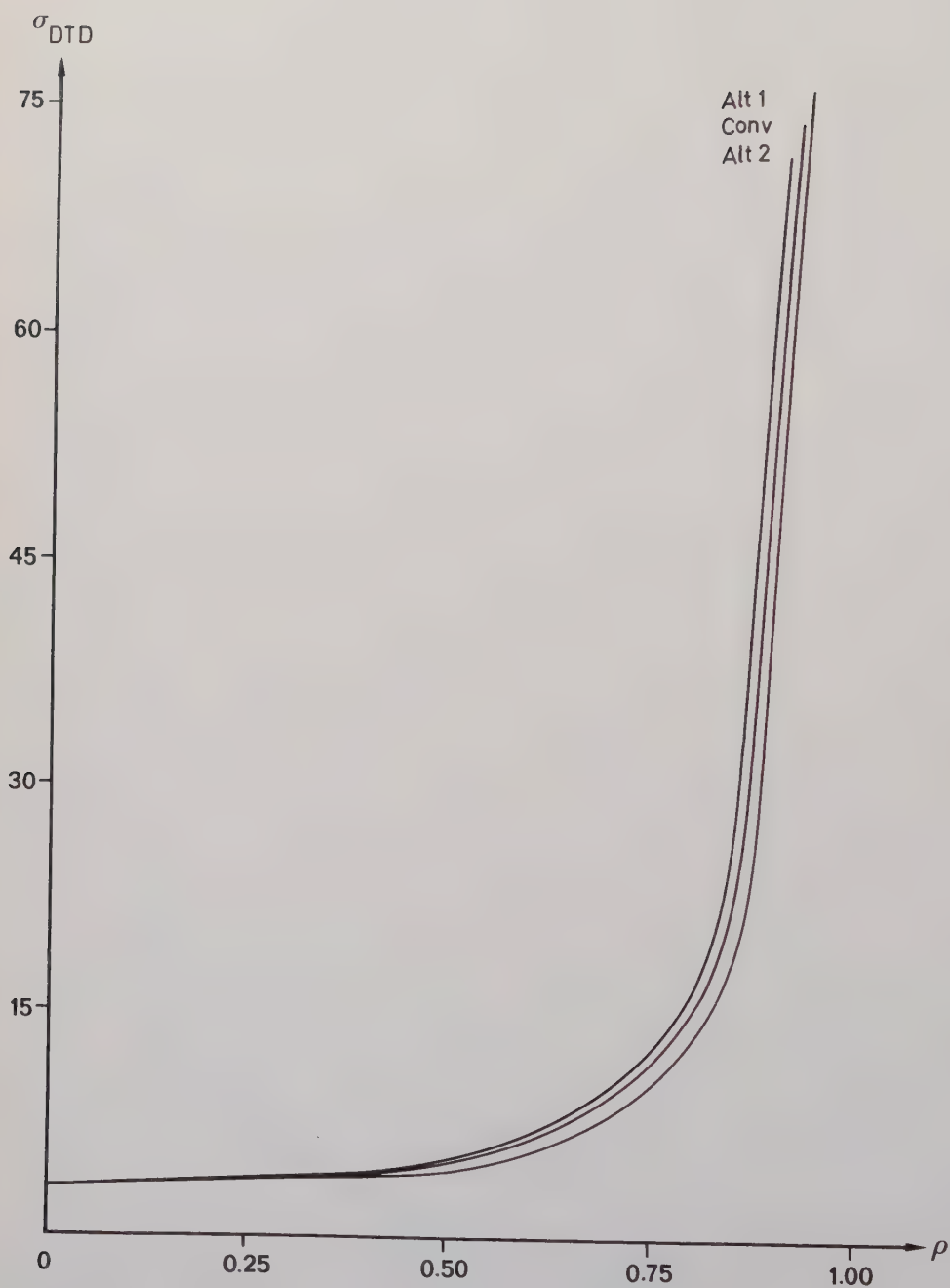


Fig 7.4

Fig. 7.4 Standard deviation of Dial Tone Delay

The difference is not great, but maintained for all the investigated measures.

If the conventional model is assumed to be the most accurate (which is not evident), then it is interesting to note that the omission of certain properties of the "real" system can act in any direction what the performance measures are concerned. Omission of the RPH would be expected to make the system more regular, and the variances would consequently be expected to decrease. But alternative model 1 overestimates the performance measures compared with the conventional approach. The polling system of the input part of the RPH gives rise to a rather irregular input traffic to the Central processor, and traffic ought to be less variable without it which could be expected to lessen the variance of for example the JBB queue length. But there are cases for which this is not true, as shown by Wolff (WOLF 82). It can therefore not be concluded that simpler models always underestimate performance measures.

Another observation to be made is that the conventional model is much too detailed for the experimental frame used here. But it is only possible to make that statement in retrospect. The exact signal sequences, the RPH and other system components could have been of importance, and some indeed were as will be shown later. The problem is to be able to gain confidence in the model without checking it against some more detailed and without making it too large. The only way to accomplish this is by collecting experiences from previous experiments and to form some kind of guidelines for the modelling. One experiment is obviously not enough for this, but the AXE simulation was picked out of many possible because it was multifaceted and still had a great deal of generality. Some other experiments, which confirm the propositions made here, will be related below.

There was a number of system properties that could be the cause of the behaviour observed from the performance measures. Apart from the structural properties, such as routing and feedback, some distinct features may have impact on the performance measures. The RP service times are deterministic, and a call will at a specific point in its progress through the system, always be processed equally long. This could be assumed to create oscillations in the system in the respect that a call that left the Central processor at one time could be predicted to arrive again at a certain future time. If the deterministic RP times were substituted by an exponentially distributed time, all determinism would be removed and measurements could show whether the deterministic times had any effect.

Figures 7.5 through 7.8 show mean and standard deviation for JBB queue length and busy periods as before. Dial Tone Delay was not measured since the mean would not be affected by the change and since the standard deviation would increase tremendously due to the exponentially distributed times.

The effects of the change are small, except for the standard deviation for busy periods. The constant RP times can therefore not be assumed to be the main cause of the observed behaviour.

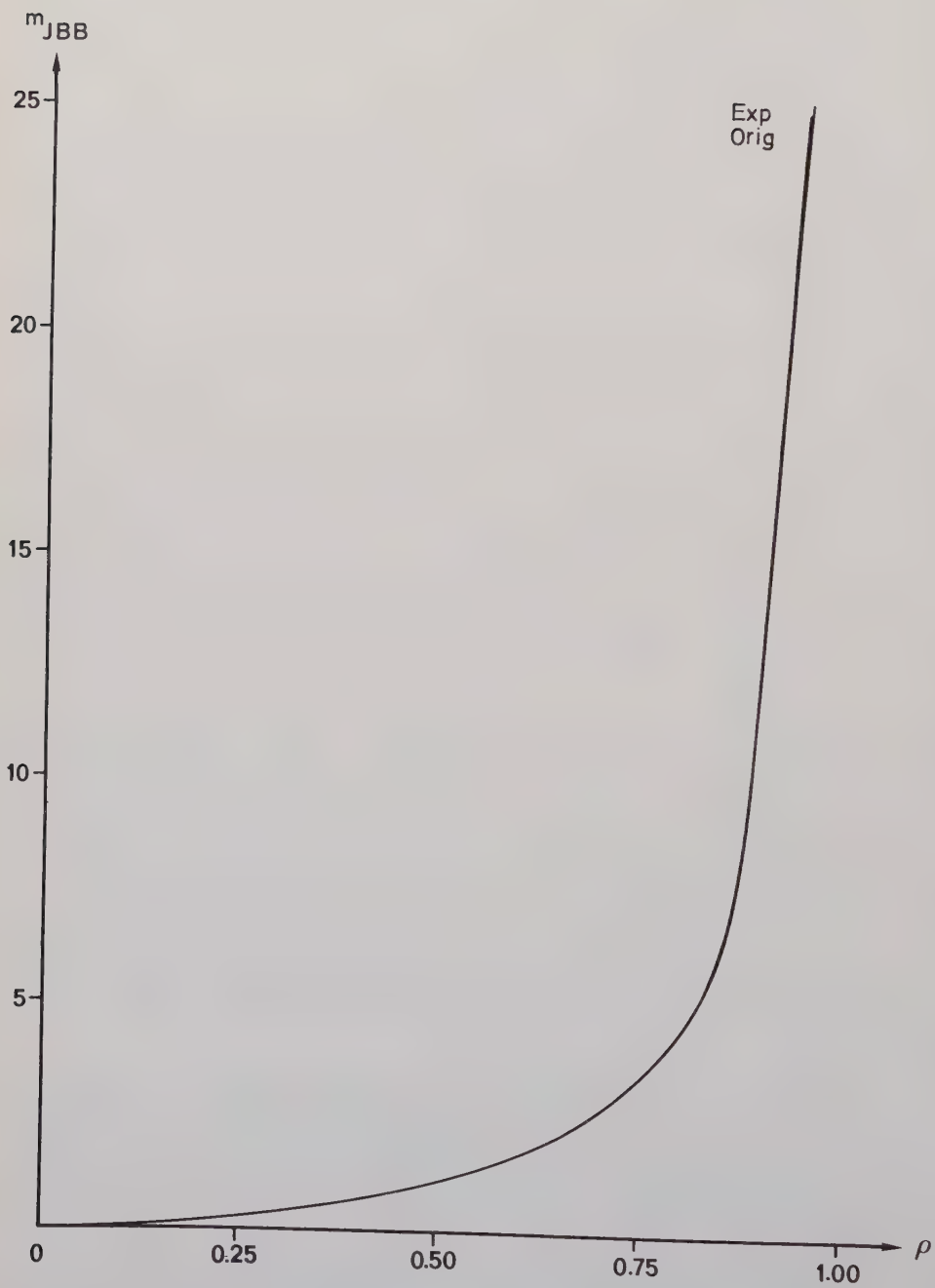


Fig. 7.5 Average queue length of JBB

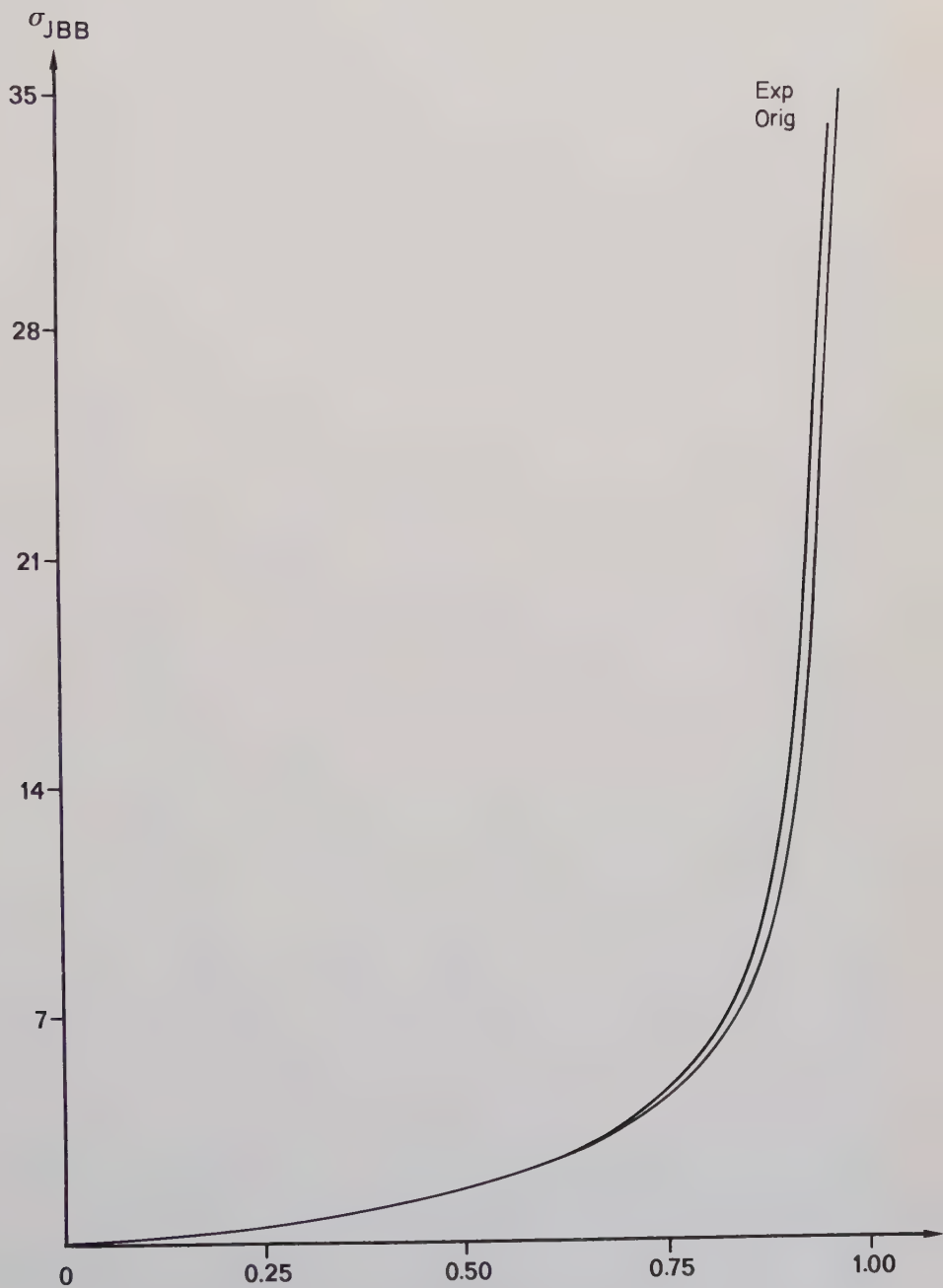


Fig. 7.6 Standard deviation of queue length of JBB

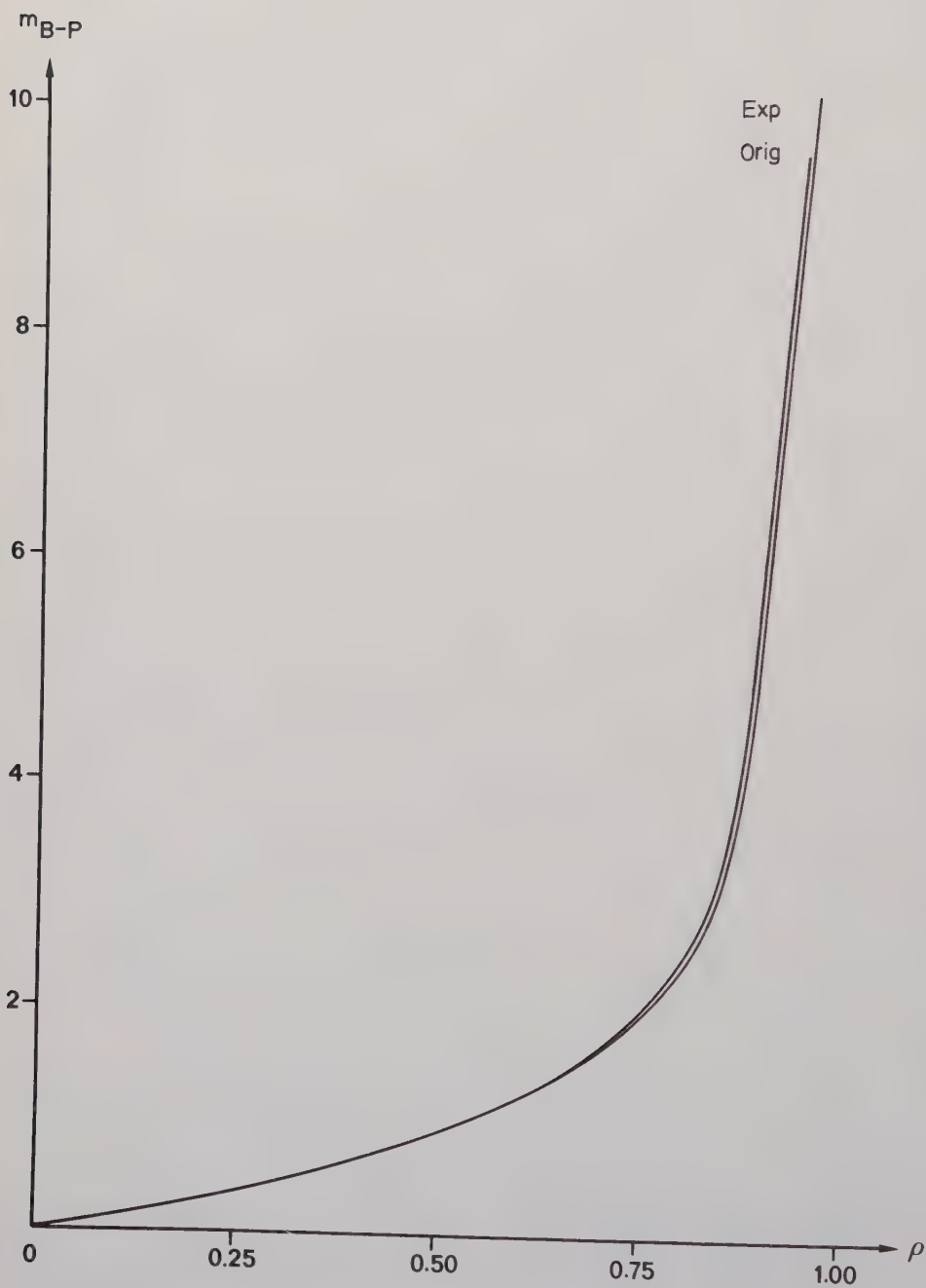


Fig. 7.7 Average length of busy periods

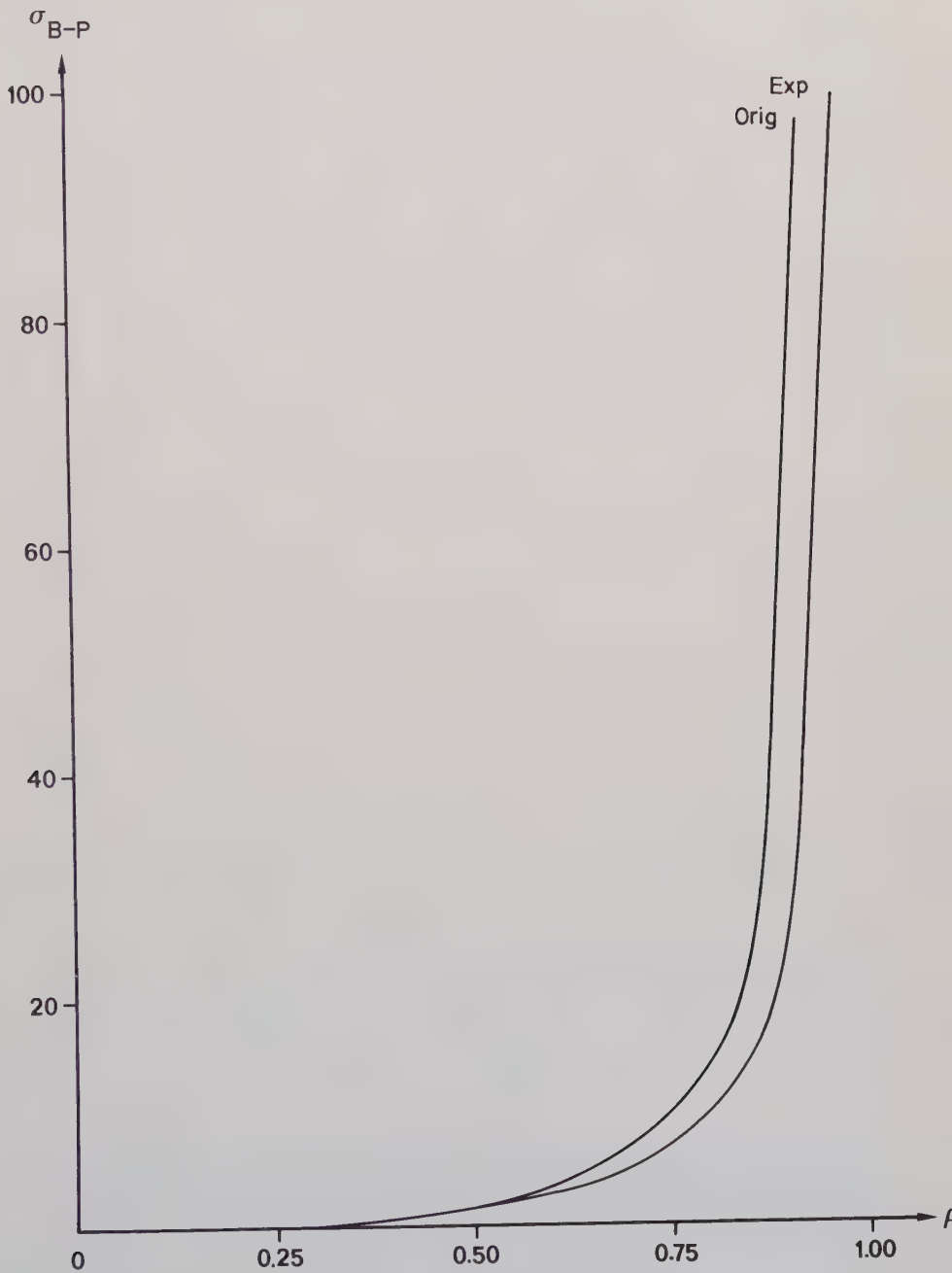


Fig. 7.8 Standard deviation of busy periods

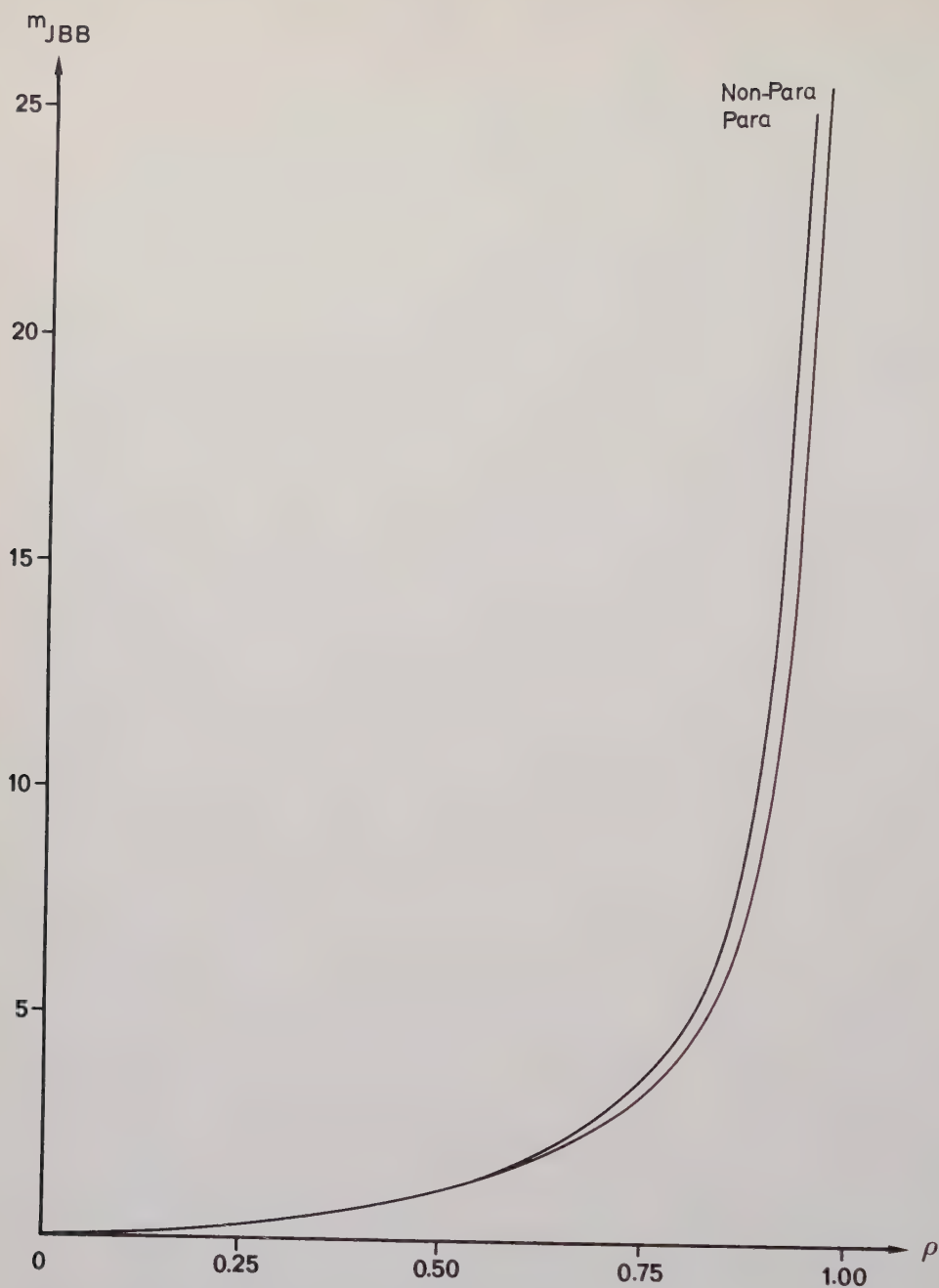


Fig. 7.9 Average queue length of JBB

An interesting property of the AXE system (and indeed of many other computerized control systems) is the existence of parallelism or forking. As a call propagates through the system it sometimes splits into two or more and the new branches continue in parallel. This behaviour emanates from the existence of independent tasks that can be executed without precedence. The same amount of work has to be done by the processors however, and from this point of view there is no difference

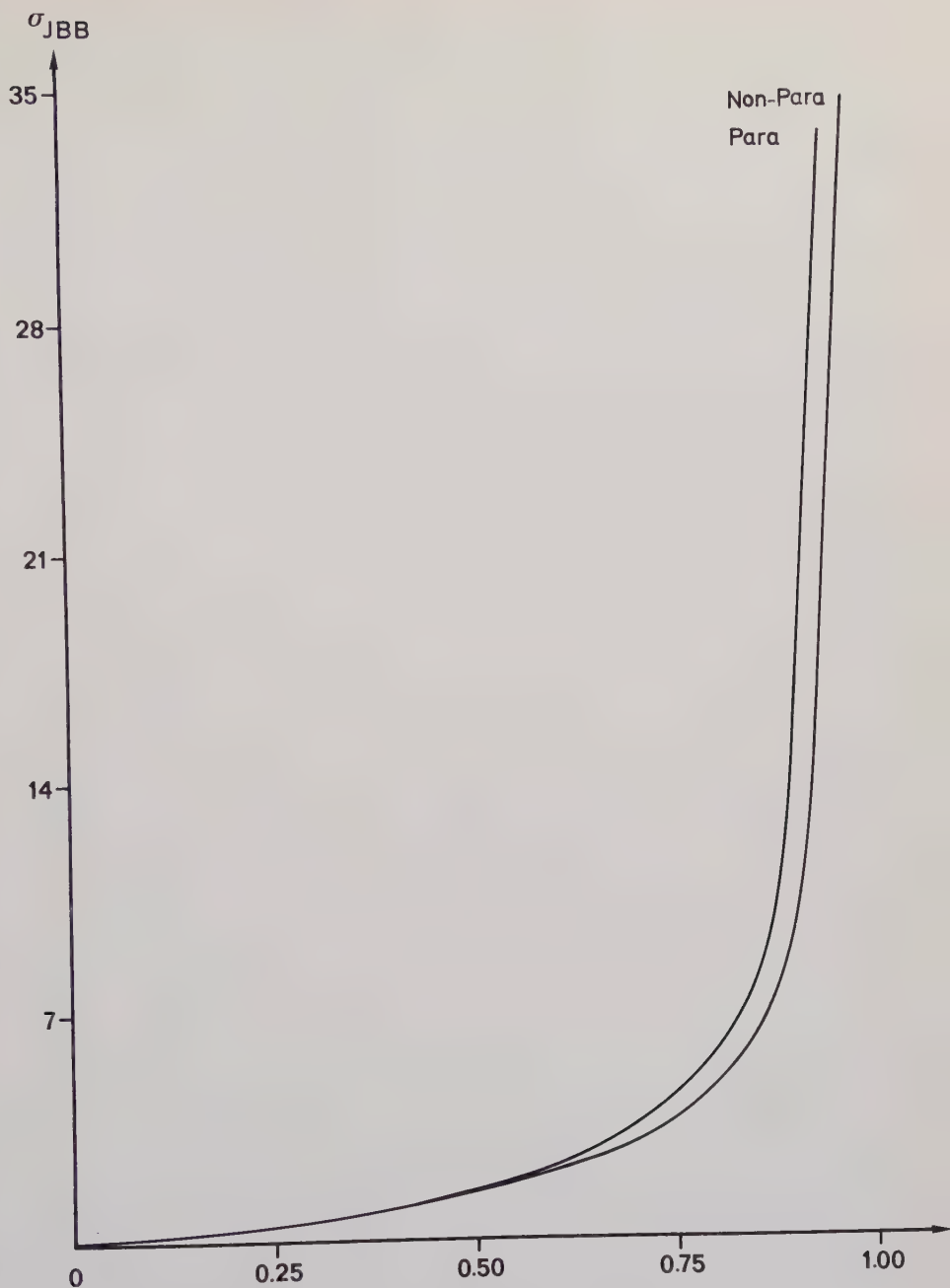


Fig. 7.10 Standard deviation of queue length of JBB

between emitting sequential signals and emitting parallel ones. The alternative model 2 was changed so that it simulated a case for which a call was set up with all signals in sequence as opposed to the original case.

Figures 7.9 through 7.13 show the effects of this for the same performance measures as above plus standard deviation for Dial Tone Delay.

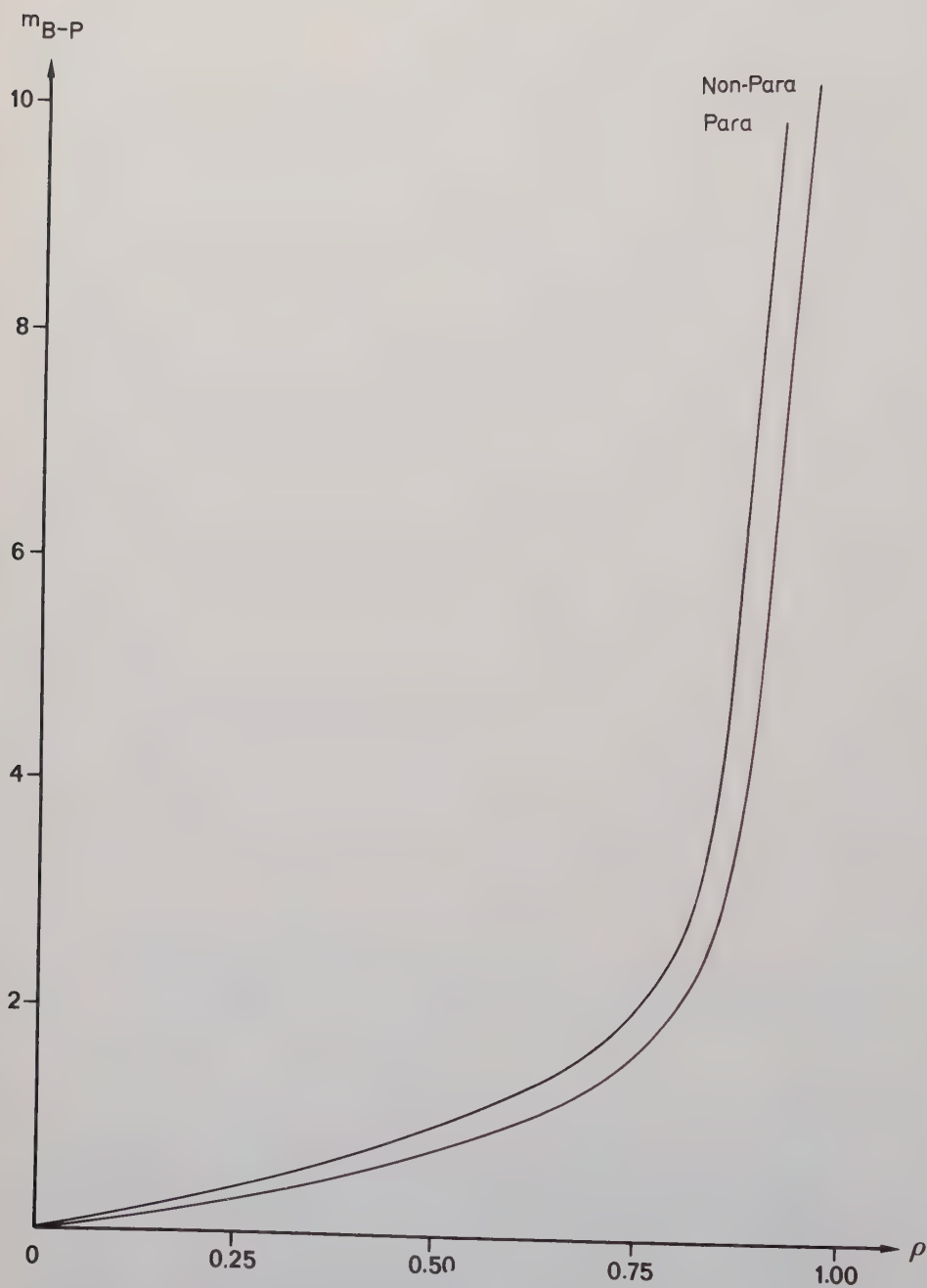


Fig. 7.11 Average length of busy periods

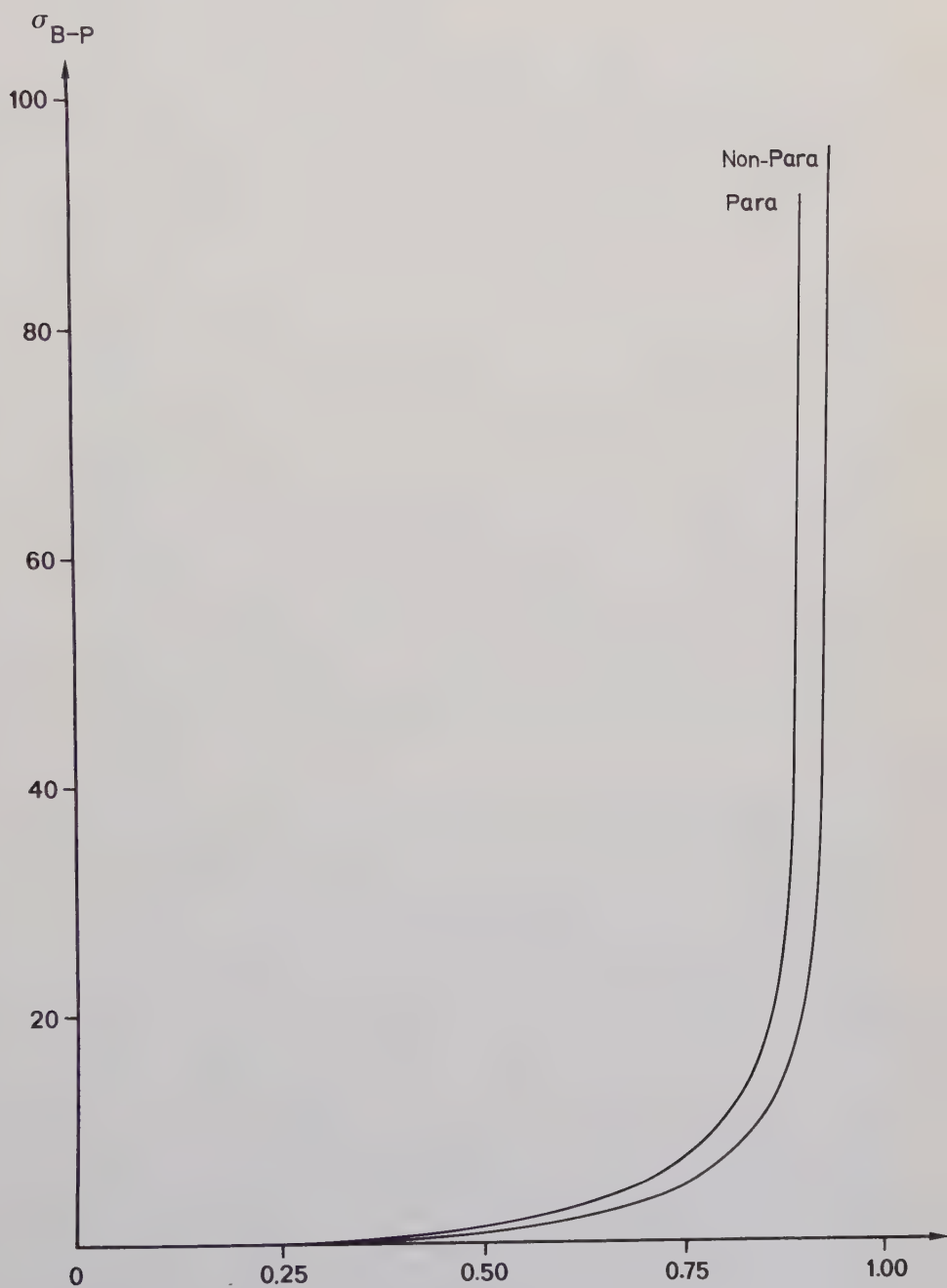


Fig. 7.12 Standard deviation of busy periods

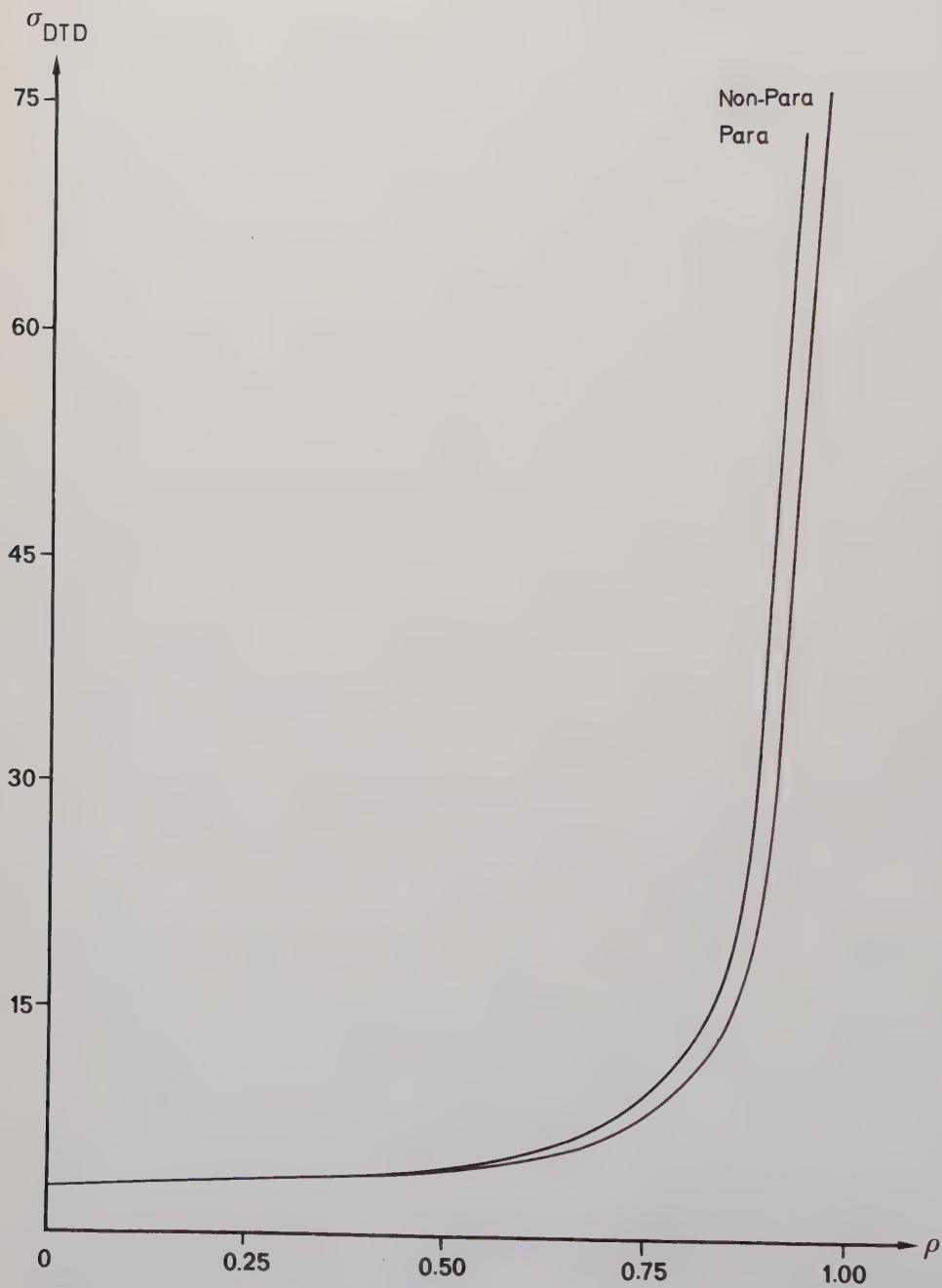


Fig. 7.13 Standard deviation of Dial Tone Delay

It can be seen from these figures that the parallelism has great impact on all shown performance measures. This may be an important observation, since the parallel tasking may not be necessary in all systems where it exists, but is used merely because it is convenient from a programming point of view. In single processor systems, like the Central processor of AXE, where the parallelism is ostensible only, nothing in particular is gained by the parallelism. The conclusion is that parallel tasking should be avoided if possible.

Much attention has been paid in analytic modelling to different service time distributions. (Less effort has been used to investigate the impact on performance measures by different arrival processes, but that is probably due to the greater complexity of such problems). Two separate investigations were performed in order to find out the importance of the service time distributions in a system of AXE's type and how the structure of the system related to the service given.

Jobs arriving at the Central processor are really requests for execution of a specific program, i.e. a piece of code. The execution of the program will take a well-defined time, and a model of the processor should therefore comprise a discrete service time distribution since the number of different jobs is finite. This discrete service time distribution happens, in the version of AXE that is used as a basis for the present investigation, to be similar to an exponential distribution. But this was of course only a coincidence, and an exchange with other types of traffic, and hence jobs, might very well have a different service time distribution. The second alternative model was run with four different service time distributions in order to investigate the possible change in performance measures. Distributions with a broad range of coefficient of variation were selected. The distributions used were Deterministic (D), Erlang-2 (E2), Exponential (Exp) and Hyper-exponential (H2). Figures 7.14 through 7.17 show how the JBB queue length and the busy periods behave with these distributions.

The differences between the curves are not extreme. It is interesting to observe however that differences occur even where not expected. In an M/G/1 system, mean of the busy-period depends on the first moment of the service time distribution and the load only. If therefore the arrival process were a Poisson process, then the average length of a busy period should not change with the service time distribution as long as the mean service time was kept constant, which it of course was here. The curves do not differ very much, and the differences could perhaps be normal statistical variations. The model has been run a substantial number of times and the difference has to be considered significant however.

There is a great difference between the deterministic service time distribution (coefficient of variation = 0) and hyper-exponential service time distribution (coefficient of variation approx. = 17). The impact on the performance measures is in spite of this not very great but rather of the same magnitude that was observed for the parallelism. A possible conclusion is then that structural properties, as feedback and parallelism may be just as important as correct service time and interarrival time distributions.

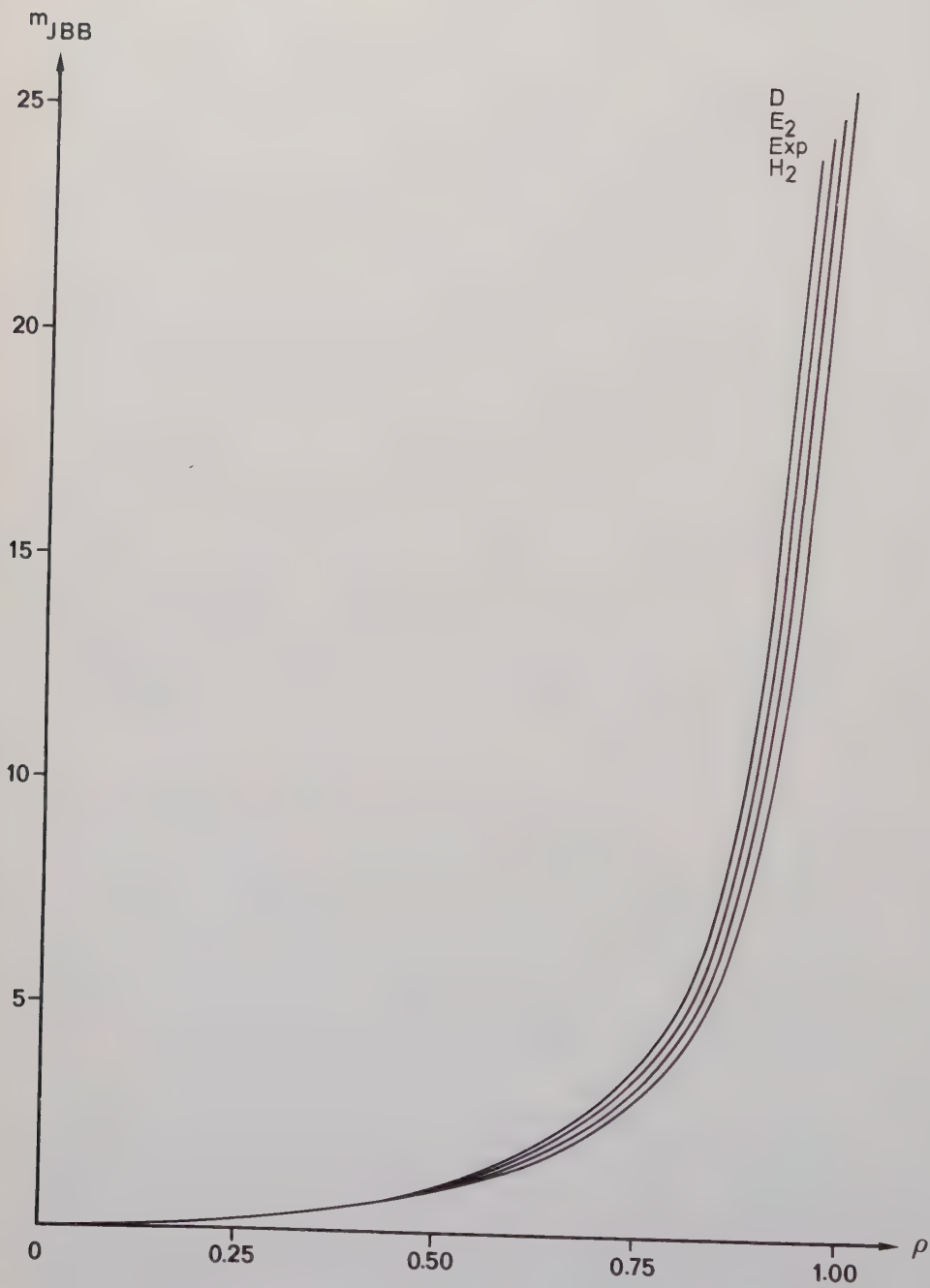


Fig. 7.14 Average queue length of JBB

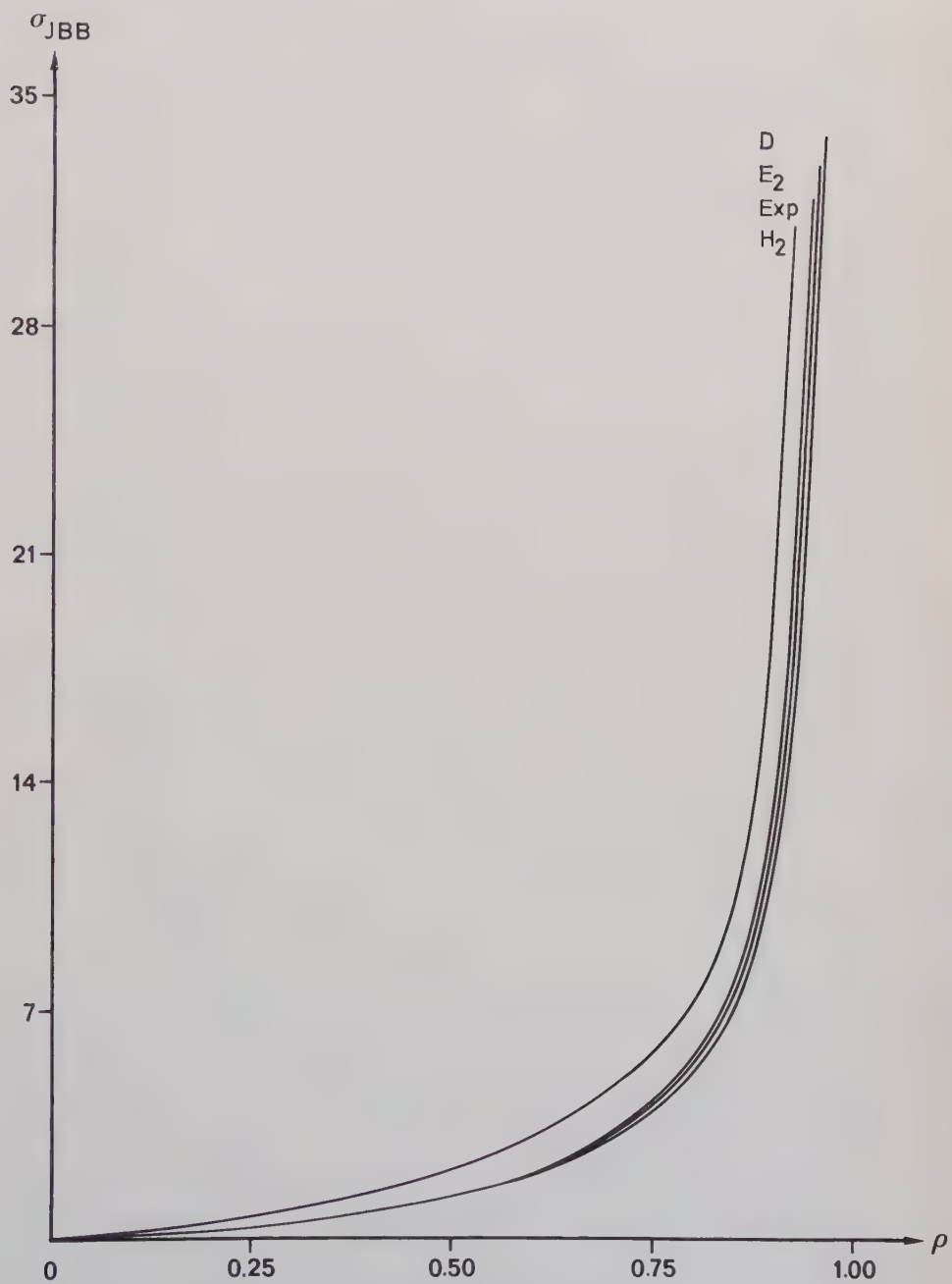


Fig. 7.15 Standard deviation of queue length of JBB

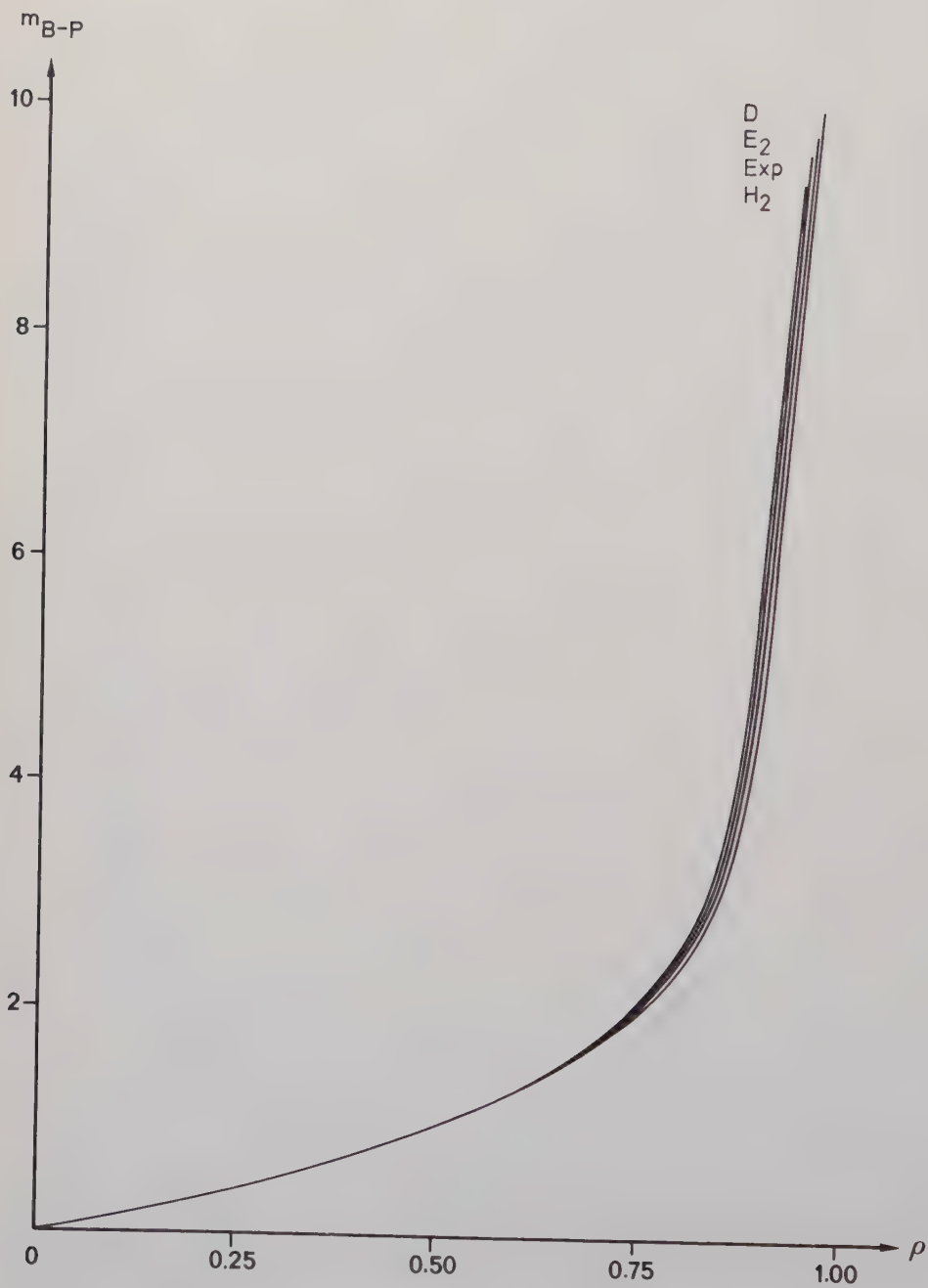


Fig. 7.16 Average length of busy periods

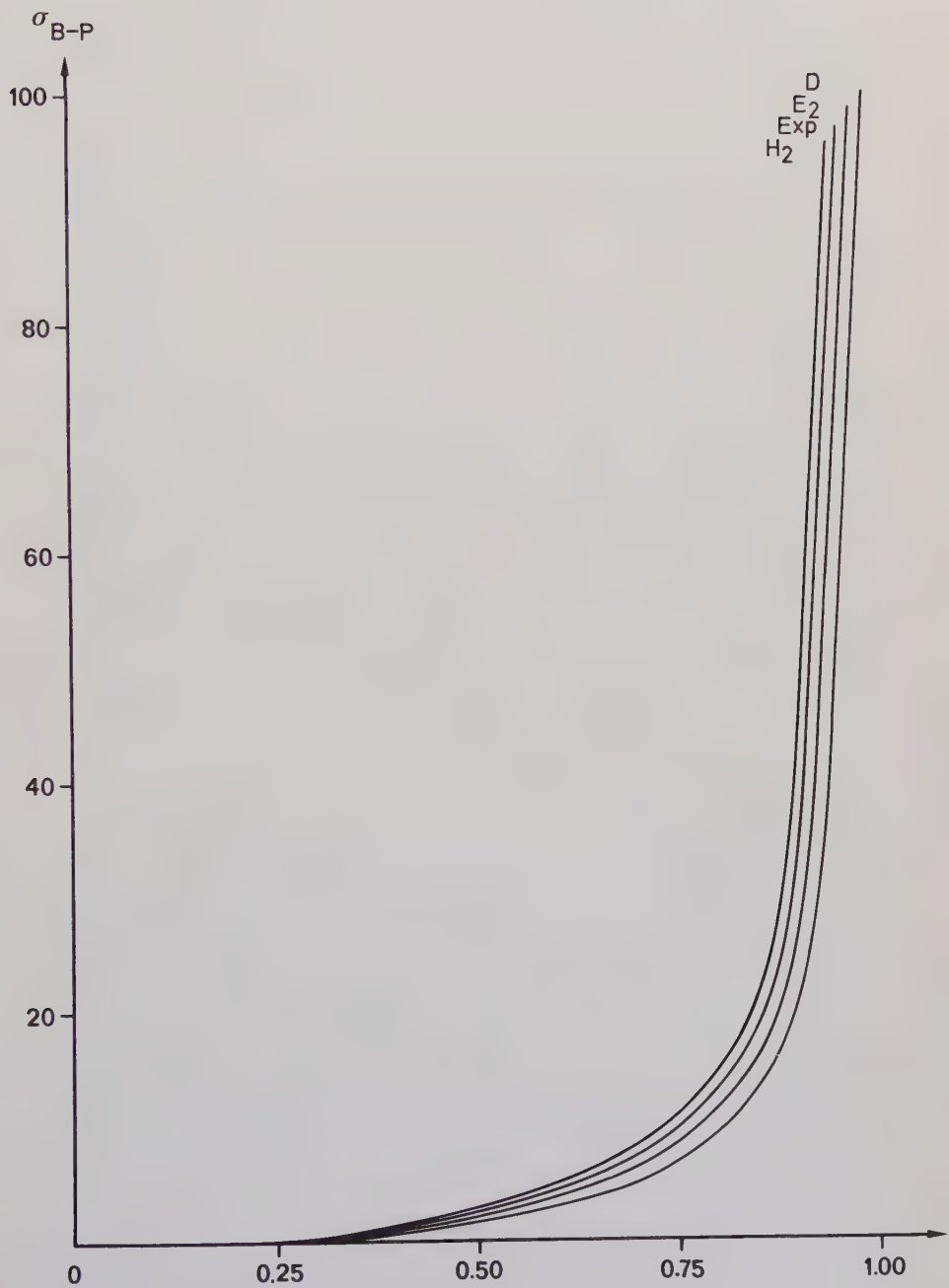


Fig. 7.17 Standard deviation of busy periods

The use of a distribution for the service times in the Central processor instead of the more correct deterministic signal sequence, could be expected to be of importance when measuring Dial Tone Delay. Since Dial Tone Delay is the time that elapses between offhook and reception of dial tone, it will in AXE imply execution of several programs in the Central processor and a number of visits to Regional processors. The total amount of code that has to be executed is predetermined and the same for all calls. The Dial Tone Delay will therefore have a lower limit, which will be equal to the time it would take to receive dial tone if there was only one call for the exchange to set up. If an exponential distribution is used instead of the exact times, no such lower limit exists, because there is a slight probability for all kinds of service times in the central processor.

An analytic model of Dial Tone Delay would have to illustrate the feedback mechanism of the Central processor as well as the limited number of turns a call makes when it arrives at the Central processor when ready at the Regional processor. Takács (TAKA 63) has made a model of a feedback system in which customers are allowed to make an infinite number of turns but in which they have a probability of leaving the system after completion of each service. He derived a Laplace transform, which is the transform for the time a customer that has made a specific number of turns, has spent in the feedback system, i.e. the response time. This model is in some respect a good description of the Central processor, since it works as a feedback system, but in reality a call cannot make more than a maximum number of turns in the processor, and is thereafter bound to leave. Takács has investigated an M/G/1 system, but it is possible to get a more realistic model of the processor if it is considered as an M/M/1 system with feedback in which customers are allowed to make at most a finite number of turns and in which they can leave at any earlier stage. An appendix at the end of the thesis contains a derivation of the response times of such a system. The calculations are brought to an appendix since they are rather complex and since they, if presented here, would only hide the result discussed below.

Dial Tone Delay consists of delay encountered at each visit at the Central processor plus delay arising from the function times in the RPs. If the function times are considered constant and load independent, only the Central processor delay is of interest since the function times will add up to a constant which can be excluded from the analysis. If each visit to the Central processor is assumed to be independent, Dial Tone Delay will consist of a sum of stochastic variables, which represent delay encountered at each visit. This type of analysis was first performed with the Takács formula, and results were compared with simulations. The results did not agree, and it was assumed that one of the reasons could be the difference between the Takács model and the Central processor in the respect that customers in the Takács model were allowed to make an infinite number of turns while they in the real system were bound to leave after a maximal number of turns.

The transform of the response time presented in appendix is rather complex. It can be numerically inverted however, and it turns out that it, with properly calibrated load, gives exactly the same results as Takács' formula. This numerical inversion is certainly not a proper proof of the

formulae giving the same results, but in addition to this it can be shown that the formulae are the same for the first four turns, which is how far it with reasonable effort is possible to carry the calculations. The transform for the fifth turn gets extremely complex and cannot easily be shown to agree with Takács' expression.

If the expression is correct, which it undoubtedly is at least for the first four turns, it is quite amazing. It means for example that a feedback system, in which all customers make exactly N turns, does not differ in respect of total response time from one in which customers are allowed to leave at any turn.

The fact that customers cannot stay indefinitely long in the more proper Central processor model, does therefore not seem to be the cause of the differences observed between analytically derived results and simulation results. Lately, improved models have shown to be able to produce more accurate results, but the models are not final and will be reported on elsewhere.

The conclusions from these investigations can be expressed thus: it is almost impossible to in advance say which properties of the system that contributes most to the performance measures. Properties, that are seemingly unimportant, may prove to have substantial impact on the performance measures, while other, and perhaps usually vital, properties can be crudely modelled without loss of accuracy. A general observation is that structural properties, as for example feedback, forking and similar features, may be just as important as the shape of the service time distributions.

Structural properties differ from a modelling point of view somewhat from properties like distributions and so forth. Modelling of structural properties is in a sense binary: the property is either incorporated or not. A distribution on the other hand can be continually improved and refined in order to increase the model validity.

8. A non-specialist approach

8.1 The simulationist's handbook?

There exists a general problem of conveying experience. How does one teach people to be creative, to find solutions to puzzling problems and to be good modellers and simulationists? Anyone, how has taught for example mathematics, knows that it is quite possible to make people understand that there are such things as primitive functions, but also that it is a much more difficult task to make people understand how to find the primitive of a specific function. It is possible to acquire great skill with some practice, but there does not seem to be a simple method of teaching how to gain the skill.

The simulationist does to a certain extent face a similar problem. There is a great number of mistakes that can be made in the pursue of a simulation task, and once these are made, it is simple for the advanced and experienced simulationist to pinpoint them and to describe how they could have been avoided. But it is much harder to set up rules or guidelines which can make the unexperienced modeller avoid the traps. It seems as if generality is the crucial obstacle. An experienced modeller can usually, if he is faced with a specific system, of which to make a model, explain why he makes certain judgements and why he considers certain properties to be of interest. But the way he approaches the task is changing, in that different systems and experimental frames require different approaches.

If some rules are set up, they will always have limited applicability. The wider area the rules try to cover, the more general they will be, and consequently less useful. It is therefore necessary to find the adequate level, at which the rules become sufficiently general and still are useful.

The bulk of this thesis has been devoted to a discussion of the modelling process, the fundamentals of queueing structures and obtainable results. It is the author's belief that the level, at which the vague guidelines of the preceding chapters reside, is the most adequate at the moment. Rules do still have to be verbal, and the introduction of a formalism, to describe simulation models in minutest detail, is ill-advised. We do currently not know enough about the modelling process to give precise rules, and a complex formalism would only hide the pertinent questions and give the models an undeserved illusion of correctness.

Two more things cannot be avoided: firstly, that guidelines have to be formulated as checklists, which contain items that possibly could be of importance, and secondly that some of the items of the checklists may seem petty to some modellers. A modeller works in his own frame of judgement, and some considerations are to him obvious while others are new and useful. A modelling scheme can therefore never fit everybody exactly, but has to appear overworked and detailed in some respects.

The next section will in a concentrated form summarise the propositions put forth in the previous chapters.

8.2 A modelling scheme

Modelling is not a straightforward process. A modelling guide can therefore not be outlined as a step by step plan, in which each step is trod only once, but more as a ladder on which one can walk up and down. But one is not allowed to advance or descend more than one rung at a time. So if the experimental frame, the presuppositions or anything else that is vital is changed, everything should be started anew. It may very well be that some steps at such a time can be taken rapidly, once we have convinced ourselves that nothing has changed since we were there the last time, but it is essential that all steps are considered.

Let us assume the existence of a conceived or "real" system which we intend to analyse. The goal is defined and we have it formulated in terms of performance measures.

- 1: Identify the experimental frame, i.e. list the performance measures of interest. Set tentatively the border between the simulation model and the outside world. Identify, without concatenating any behaviour in a "smart" or "efficient" manner, possible processes of the system.

Comments: This step is indeed tentative, and should be possible to alter at any later stage. The modelling of a system should be viewed as a process with feedback, in which the modelling itself can bring forth insights which may change the presuppositions, and thus the decisions made in the modelling process. It is therefore necessary to be receptive to such changes and not to hesitate if remodelling seems to be necessary. A lot depends on this step and it should therefore be carefully considered.

- 2: Perform, for all the tentatively defined processes of step 1, a Deterministic Analysis as described in chapter six.

Comments: This step is probably one of the most laborious. It may require a lot of research to acquire all the information necessary for the Deterministic Analysis, but it usually pays off in the end when input data to the simulation is to be collected. The major part of that work has already been done in the Deterministic Analysis, and be therefore sure to save the information brought up.

- 3: Illustrate the outside world by a number of generators. These generators should create jobs and enter them into the structure at some appropriate point.

Comments: It is, with the approach employed here, the service centres that are the active components of the simulation. This means that the generators create only some passive data structure and make the appropriate service centre aware of the arrival (if necessary). It is common that the generators contain only one scheduling statement, but more elaborate structures are possible. A point to consider, however, is the

time dependence of the generators. The load of the network depends directly on the intensity of the generators. To be able to make any useful measurements requires the system to behave predictably. If the load for example differs during the period of measurement, and queue lengths are measured, it becomes very hard to interpret the results. The generators should therefore not be made too elaborate unless varying load is of specific interest.

- 4: At this stage, all information necessary for an Operational Analysis is available. From this, it is possible to deduce the approximate load of the service centres. Do so.

Comments: The approximate arrival rate produced by the generators and the service times given by the tentative structure of the processes and the Deterministic Analysis will give the load of the individual processes. The step may involve estimation of routing probabilities produced by routing, feedback or internal scheduling, but the final result should be a figure ranging from zero to unity.

- 5: Consider for omission from the simulation model processes that are lightly loaded and which are of no particular interest.

Comments: This step has to be conducted with care, since there are cases when lightly loaded processes may not be excluded. Obvious are for example cases when the process itself is of interest and when the delay caused by the component is long (inspite of it being lightly loaded). The latter case can sometimes be handled by letting the preceding process delay the job a time equal to the service time of the following process. If routing is complex, this may be unfeasible however. Another reason for keeping the process in the model in spite of it being lightly loaded is that the process may give rise to extreme variance of service times. If for example the service times depend on the state of the system in some way, so that service times alter between two opposite extremes, there may be reason to keep the process. Other similar cases may come up.

- 6: Concatenate scheduling statements obtained in step 2 until an acceptable balance is obtained. Consider especially the placement of random number generators.

Comments: The Deterministic Analysis does usually bring the analysis down to a level which is too low to be modelled in detail. Behaviour must therefore be concatenated in order to increase the time level and to decrease the level of detail. The concatenation should not be done arbitrarily however. It should start at the lowest level, i.e. in the fastest process, and move upwards. It is obvious that internal behaviour of specific interest cannot be concatenated in this way, but has to be left as it is.

Another point to consider is the placement of delays governed by distributions. There is a general risk that decisions illustrated as random events, are placed at too high a level in the simulation model. It may sometimes be impossible to lower the level, but when possible it should be done. To make the point a bit more clear, let us elaborate somewhat.

Assume that events at some low level of the simulation depend on events at a higher level, and that the events at the high level are governed by a stochastic distribution. Do also assume that we are interested in measuring events at the low level in order to find out some performance mea-

sure. It is then possible that a considerable number of events occur at the low level, and that we consequently can collect large amounts of data, while only few events occur at the high level. Depending on the start value of the random number generator that generates the sequence of numbers at the high level, things may look quite different at the low level. One run may differ considerably from another, and unless replicative runs are made, which sometimes are hard to administrate, results will be rather ambiguous. If only a few random numbers are drawn at the high level, different runs may show entirely different performances. Are we not observant, the effects will not present themselves at the low level.

It is at this stage of the modelling process usually possible to estimate the running times and the approximate size of the program. It is advisable to do so before proceeding.

- 7: Consult the checklist at the end of chapter 6, and check whether any of the listed behaviour is present in the system that is being modelled. If so, investigate the possible impact these properties could have on the performance measures.

It is interesting to observe that so far no programming has taken place. All work till this stage is pure paperwork, and should not take too much time. This is important, since the modelling may have provided insight which requires some of the above modelling steps to be reconsidered.

Many steps remain at this stage, but it is beyond the scope of this text to deal with them. From here on there is not much difference between simulation and ordinary programming, for which there are abundant literature.

9. Conclusion

Experiences gained when simulation experiments are pursued do in most cases not appear as to be systematic. All simulation experiments are unique, and do still have something in common. Even for very limited areas of application, as for example queueing structures, new situations and combinations always appear. New technical systems are invented and new solutions to old problems are continually presented. The simulationist who is asked to model such systems does therefore over and over again have to come up with new models, and it is uncommon that already programmed models can be used. From this multitude of models, programs, simplifications and conclusions a pattern can eventually be observed. In the modelling process, recognition comes up, and the modeller "knows" that a specific problem calls for a specific solution. But how does he "know" that? What has convinced him of the correctness of his decisions?

Bertrand Russell once said that one of the problems of democracy is that the stupid are so self-confident and that the wise are so uncertain. The same can perhaps be said about simulation modelling. Simulation is seemingly simple and in that sense a bit seductive. The unexperienced modeller may, usually by himself, be lead to believe that because his first simple models were so successful, he is able to build larger and more complex models in a never ending progression. Reality is a good master however, and the modeller does quite soon realise that modelling is more complex than he at first thought. He is about to become a wise modeller. But he is also becoming susceptible to the great risk of the insight: to think that simulation is of no use and that the possible errors are so many that no model can be expected to be correct, and to think that all simulations indeed are unique and that nothing learned from one experiment can be used in some other.

That can of course be true, but it is this author's opinion it is not. There are patterns; "reality" does indeed appear as regular and there are phenomena in this world which have much in common and which can be treated in almost the same way and analysed with almost the same methods. But the kinship is not easily grasped. It has to be described in rather vague and imprecise terms and cannot be formalised in the same way as more narrow subjects can be.

Another question is how much we have to experience, how many systems we have to analyse and how many mistakes we have to make before we are allowed to set up rules for other modellers to follow. This thesis has presented one medium sized example and shown how simplifications could be made without great loss of accuracy within the experimental frame. A not far-fetched objection could be that the modelling scheme and the propositions put forth are too detailed and specific to be based on a single example. This is quite true. The example is only one out of many possible however, and was selected only because it showed some interesting properties that are likely to be found in systems of similar kind. Several other simulations have, in support of the ideas, been performed over the years, and they have all developed along lines not unlike

the one here presented. The guidelines presented above are the smallest common divisor of these experiments, and it is obvious that there are cases for which they do not hold, but also that some other group of systems could have an even smaller common divisor. Presenting guidelines of the above kind is always a question of belief. One either thinks they are sensible or one does not, and it is impossible to prove that someone who does not is wrong.

The additional experiments that talk in favour of the proposed guidelines have been left out since they would in much resemble the one described, and also because descriptions of simulation experiments are tedious both to read and make. But there are more things than additional descriptions that have been left out and which could have defended a place in an investigation of this sort. The proposed modelling scheme assumes the modeller to be in possession of some description language which can be used in the early stages of the modelling process. Not much attention has been paid to these issues, although important and vital to a successful pursuit of a simulation task. Much can be said about how to organize input data, data structures in the simulation program and how to write the simulation program, but such remarks become even pettier than a modelling scheme and do not belong in this context. But they are all the same important and should not be neglected. Work done in parallel with the investigations presented here have dealt more specifically with these issues and will hopefully be presented in the near future.

It is customary to say something about directions for future research at the end of a thesis. This issue does in reality split into two: firstly that it is important to spread the already available knowledge to a wider group of simulation practitioners, and secondly that we still know too little about modelling and simulation. What the first issue is concerned, it is obvious that we have to get away from the rather primitive simulation techniques in use to-day and leave the FORTRAN based simulation languages. There are better ways to write simulation programs and describe simulation models than messy FORTRAN programs.

As for the second issue, much is still hidden in the mist. We do indeed not know what actually happens when a model is formed. Our ability to grasp complex systems is very limited and development of more versatile support mechanisms is a extremely important research objective. But this is probably not enough. A collaboration of various disciplines, as psychology, mathematics and stochastic theory, can perhaps be a fruitful approach. It is essential to understand what we mean by accurate models and to form some kind of measure for this. The most common measure in use to-day — how detailed the model is — is not a good measure. A better measure has to be based on how great impact a specific property of a system has on the performance measures we are interested in.

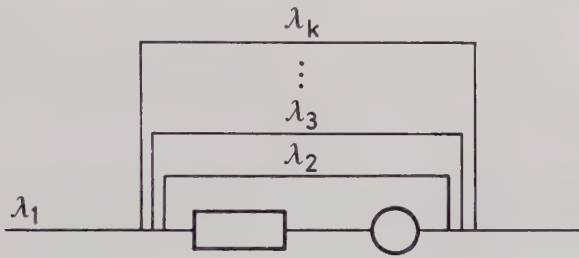
Another objective is to make simulation suited for performance estimation at very early stages of system development. To accomplish this, modelling has to be directed away from the pure copying of detailed behaviour and narrowed to a more "of the shelf" simulation technique. This should not mean, however, that crucial modelling steps are automated and brought away from the modeller, but rather that the modeller is aided as much as possible without letting the computer make the important decisions.

APPENDIX

Consider an M/M/1-queue with feedback, in which customers are allowed to receive at most k services and in which customers can leave after having received $1, 2, \dots, k-1$ services. The probability that a customer requests another service after having obtained its j :th service, is assumed to be α_j , $j = 1, 2, \dots, k$ and $\alpha_j = 0$, $j \geq k$.

The system can be looked upon as an open queueing network with class switching, in which the probability of switching to a higher class is α_j if the present class is j . Customers who enter the system are said to belong to class 1. The arrival rate from the Poisson source is assumed to be λ .

The following figure illustrates the system



If we assume the average service requests to be the same for all classes and equal to $\frac{1}{\mu}$, we have that

$$\lambda_1 = \lambda$$

$$\lambda_2 = \alpha_1 \lambda$$

$$\lambda_k = \lambda \cdot \prod_{r=0}^{k-1} \alpha_r, \text{ where by definition } \alpha_0 = 1.$$

Queueing Network Theory does then tell us that

$\text{Prob}(n_1 \text{ customers in the first class, } n_2 \text{ in the second, } \dots, n_k \text{ in the } k\text{th; at random instants})$ is

$$P(n_1, n_2, \dots, n_k) = \frac{n!}{n_1! n_2! \dots n_k!} \prod_{r=1}^k (\rho_r)^{n_r} \left(1 - \sum_{r=1}^k \rho_r\right)$$

where

$$\rho_r = \frac{\lambda}{\mu} \prod_{i=0}^{k-1} \alpha_i \quad \text{and} \quad n = n_1 + n_2 + \dots + n_k$$

We also have that

$$\begin{aligned} P(n \text{ customers in the system at a random instant}) &= \\ &= \rho^n (1-\rho) \quad ; \quad \rho = \sum_{r=1}^k \rho_r \end{aligned}$$

and thus

$$\begin{aligned} p(n_1, n_2, \dots, n_k | n) &= \left(\frac{n!}{n_1! n_2! \dots n_k!} \prod_{r=1}^k (\rho_r)^{n_r} (1-\rho) \right) / (\rho^n (1-\rho)) \\ &= \frac{n!}{n_1! \dots n_k!} \prod_{r=1}^k \left(\frac{\rho_r}{\rho} \right)^{n_r} \end{aligned}$$

In the sequel we shall also use the transform

$$E\{z_1^{n_1} z_2^{n_2} \dots z_k^{n_k}\} = Q(z_1, z_2, \dots, z_k) = \sum_{n=0}^{\infty} P(n) \sum_{n_1+n_2+\dots+n_k=n}$$

$$P(n_1, n_2, \dots, n_k | n) z_1^{n_1} z_2^{n_2} \dots z_k^{n_k} = \sum_{k=0}^{\infty} P(n) \left(\frac{\rho_1}{\rho} z_1 + \dots + \frac{\rho_k}{\rho} z_k \right)^n$$

$$= \sum_{n=0}^{\infty} \rho^n (1-\rho) \left(\frac{\rho_1}{\rho} z_1 + \dots + \frac{\rho_k}{\rho} z_k \right)^n =$$

$$= \frac{1-\rho}{1 - (\rho_1 z_1 + \rho_2 z_2 + \dots + \rho_k z_k)}$$

Let us consider a tagged customer with reference to the figure below.

Denote by T_n his time in the system up to and including the n :th service.

Let

$$\tau_1 = T_1 ; \tau_n = T_n - T_{n-1} ; n = 2, 3, \dots, k$$

and

$$\tilde{r}_{ij} = (r_{i1}, r_{i2}, \dots, r_{ij})$$

where

r_{mn} = number of class n customers present at the "tagged" customer's arrival who will leave after having received another m services.

and let

a_i = number of customers who arrived during τ_i .

Let furthermore

s_{mn} = number of customers arriving in τ_m that leave the system in τ_{m+n}

Also we introduce the notations

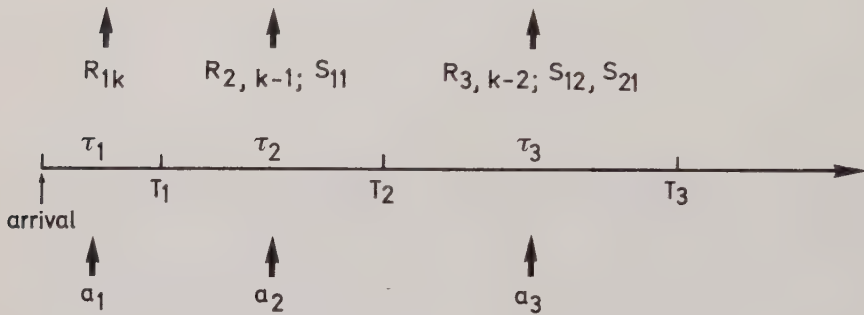
$$R_{ij} = r_{i1} + r_{i2} + \dots + r_{ij}$$

and

$$n = n_1 + n_2 + \dots + n_k,$$

the total number of customers present at the arrival of the "tagged" customer.

The notation is illustrated in the figure below



Clearly we have

$$E\{e^{-sT_1} | \tilde{n}\} = B(s)^{1+n}$$

where

$$B(s) = \int_0^{\infty} e^{-st} \mu e^{-\mu t} dt = \frac{\mu}{s+\mu}, \text{ i.e. the Laplace transform of the service time distribution}$$

and further that

$$\begin{aligned} E\{e^{-sT_2} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1\} \\ = e^{-s\tau_1} \cdot [B(s)]^{n+1-R_{1,k}+a_1} \end{aligned}$$

where $\tilde{n} = (n_1, n_2, \dots, n_k)$

$$E\{e^{-sT_3} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}, \tau_2, a_2, s_{11}\}$$

$$= e^{-s(\tau_1 + \tau_2)} [B(s)]^{n+1-R_{1,k}-R_{2,k-1}+a_1+a_2-s_{11}}$$

$$E\{e^{-sT_4} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}, \tau_2, a_2, \tilde{r}_{3,k-2}, \tau_3, a_3, s_{11}, s_{12}, s_{21}\}$$

$$= e^{-s(\tau_1 + \tau_2 + \tau_3)} [B(s)]^{n+1-R_{1,k}-R_{2,k-1}-R_{3,k-2}+a_1+a_2+a_3-s_{11}-s_{12}-s_{21}}$$

etc.

Let us start with the calculation of $E\{e^{-sT_2}\}$ to show the technique. Clearly the conditions are interdependent and cannot be removed in arbitrary order.

First, remove the condition on a_1 . Since the arrivals form a Poisson process, we have that

$$P(\ell \text{ arrivals in } \tau \text{ time units}) = \frac{(\lambda\tau)^\ell}{\ell!} e^{-\lambda\tau}$$

and thus

$$E\{e^{-sT_2} | \tilde{n}, \tilde{r}_1, \tau_1\} = \sum_{\ell=0}^{\infty} e^{-s\tau_1} [B(s)]^{n+1-R_{1,k}+\ell} \frac{(\lambda\tau_1)^\ell}{\ell!} e^{-\lambda\tau_1}$$

$$= e^{-\tau_1(s+\lambda)} [B(s)]^{n+1-R_{1,k}} \cdot \sum_{\ell=0}^{\infty} \frac{(\lambda\tau_1 B(s))^\ell}{\ell!}$$

$$= [B(s)]^{n+1-R_{1,k}} \cdot e^{-\tau_1(s+\lambda(1-B(s)))}$$

To remove the condition on $\tilde{r}_{1,k}$ we have to consider one class of customers at a time, but the generality of the calculations will soon be apparent.

For all but the highest class there are n_i ($1 \leq i \leq k-1$) potential departures to consider. Each of the n_i customers has a probability $1-\alpha_i$ of departing after a service completion.

The element of $\tilde{r}_{1,k}$ that corresponds to n_i , i.e. $r_{1,i}$, can consequently range from zero to n_i . Removal of the condition on $r_{1,i}$ will lead to a sum over all possible values of $r_{1,i}$ and we have

$$\sum_{r_{1,i}=0}^{n_i} \binom{n_i}{r_{1,i}} \beta_i^{r_{1,i}} \alpha_i^{n_i - r_{1,i}} \cdot [B(s)]^{-r_{1,i}} =$$

$$= \left(\frac{\beta_i}{B(s)} + \alpha_i \right)^{n_i}$$

where $\beta_i = (1-\alpha_i)$

We thus find that

$$E\{e^{-sT_2} | \tilde{n}, \tau_1\} = [B(s)]^{n+1} \prod_{i=1}^k \left(\frac{\beta_i}{B(s)} + \alpha_i \right)^{n_i} \cdot e^{-\tau_1(s + \lambda(1-B(s)))}$$

The sum above is not valid for $i=k$ since all customers who belong to class k are bound to leave after receiving their next service. A special case therefore apply for $i=k$. But since α_j is defined to be zero for $j \geq k$, we may formally use the resulting expression for k as well. The last factor of the product above collapses into $[1/B(s)]^{n_k}$ which is cancelled by the corresponding term of $[B(s)]^n$, as it should. This property of the expressions will be used in the following.

To remove the condition on τ_1 we have to know the density function of τ_1 . But since τ_1 has the same distribution as T_1 , we directly find that

$$E\{e^{-s\tau_1}\} = [B(s)]^{n+1}$$

If we remove the condition on τ_1 we find that

$$E\{e^{-sT_2} | \tilde{n}_k\} = [B(s)]^{n+1} \prod_{i=1}^k \left(\frac{\beta_i}{B(s)} + \alpha_i \right)^{n_i} \cdot [B(s+\lambda(1-B(s)))]^{n+1}$$

$$= B(s) \cdot B(s+\lambda(1-B(s))) \cdot \prod_{i=1}^k (\beta_i + \alpha_i B(s))^{n_i} \cdot [B(s+\lambda(1-B(s)))]^n$$

since $\alpha_k = 0$

and hence

$$= B(s) \cdot B(s+\lambda(1-B(s))) \cdot \prod_{i=1}^k [(\beta_i + \alpha_i B(s)) B(s+\lambda(1-B(s)))]^{n_i}$$

which yields

$$E\{e^{-sT_2}\} = B(s) B(s+\lambda(1-B(s))) \sum_{n=0}^{\infty} \sum_{n_1+n_2+\dots+n_k=n} \prod_{i=1}^k [(\beta_i + \alpha_i B(s)) \cdot B(s+\lambda(1-B(s)))]^{n_i} \cdot P(n_1, n_2, \dots, n_k)$$

and

$$E\{e^{-sT_2}\} = B_1(s) B_2(s) Q(t_1, t_2, \dots, t_k)$$

where

$$B_1(s) = B(s)$$

$$B_2(s) = B(s+\lambda(1-B(s)))$$

and where

$$t_i = \beta_i B_2(s) + \alpha_i B_1(s) B_2(s) \quad 1 \leq i \leq k$$

If we now look at $E\{e^{-sT_3}\}$ we find that we can proceed in the same manner.

$$E\{e^{-sT_3} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}, \tau_2, a_2, s_{11}\} =$$

$$= e^{-s(\tau_1 + \tau_2)} [B(s)]^{n+1-R_{1,k}-R_{2,k-1}+a_1+a_2-s_{11}}$$

Removal of the condition on a_2 yields

$$E\{e^{-sT_3} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}, s_{11}, \tau_2\} =$$

$$= e^{-s(\tau_1 + \tau_2)} [B(s)]^{n+1-R_{1,k}-R_{2,k-1}+a_1-s_{11}} \cdot e^{-\lambda\tau_2(1-B(s))}$$

s_{11} can range from zero to a_1 , and the probability for the customers, who arrived while the "tagged" customer awaited his service, to leave after completion of a single service is α_1 .

We therefore have to compute the following sum to remove the condition on s_{11}

$$\sum_{s_{11}=0}^{a_1} \binom{a_1}{s_{11}} \beta_1^{s_{11}} \alpha_1^{a_1-s_{11}} [B(s)]^{-s_{11}} = \left(\frac{\beta_1}{B(s)} + \alpha_1 \right)^{a_1}$$

and thus

$$E\{e^{-sT_3} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}, \tau_2\} =$$

$$= e^{-s(\tau_1 + \tau_2)} [B(s)]^{n+1-R_{1,k}-R_{2,k-1}+a_1} \cdot$$

$$\cdot e^{-\lambda\tau_2(1-B(s))} \left(\frac{\beta_1}{B(s)} + \alpha_1 \right)^{a_1}$$

$$= e^{-s(\tau_1 + \tau_2)} [B(s)]^{n+1-R_{1,k}-R_{2,k-1}} \cdot$$

$$\cdot e^{-\lambda\tau_2(1-B(s))} (\beta_1 + \alpha_1 B(s))^{a_1}$$

The density function of τ_2 has the Laplace transform

$$E\{e^{-s\tau_2}\} = [B(s)]^{n+1-R_1, k+a_1}$$

and removal of the condition on τ_2 therefore yields

$$\begin{aligned} E\{e^{-sT_3} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}\} &= \\ &= e^{-s\tau_1} [B(s) \cdot B(s+\lambda(1-B(s)))]^{n+1-R_1, k} [B(s+\lambda(1-B(s)))]^{a_1} \\ &\quad \cdot [B(s)]^{-R_2, k-1} (\beta_1 + \alpha_1 B(s))^{a_1} \end{aligned}$$

Next we consider the condition on $\tilde{r}_{2,k-1}$. The elements of $\tilde{r}_{2,k-1}$ can range from zero to $n_i - r_{1,i}$, ($1 \leq i \leq k-1$), but since $\alpha_j = 0$ for $j > k$, we can write down a more general expression and get the following sum

$$\begin{aligned} &\sum_{r_{2,i}=0}^{n_i - r_{1,i}} \binom{n_i - r_{1,i}}{r_{2,i}} \beta_{i+1}^{r_{2,i}} \alpha_{i+1}^{n_i - r_{1,i} - r_{2,i}} [B(s)]^{-r_{2,i}} \\ &= \sum_{r_{2,i}=0}^{n_i - r_{1,i}} \binom{n_i - r_{1,i}}{r_{2,i}} \left(\left(\frac{\beta_{i+1}}{B(s)} \right)^{r_{2,i}} \cdot \alpha_{i+1}^{n_i - r_{1,i} - r_{2,i}} \right) \\ &= \left(\frac{\beta_{i+1}}{B(s)} + \alpha_{i+1} \right)^{n_i - r_{1,i}} \quad \forall i \end{aligned}$$

And thus

$$E\{e^{-sT_3} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1\} =$$

$$\begin{aligned}
&= e^{-s\tau_1} [B(s) \cdot B(s+\lambda(1-B(s)))]^{n+1-R_1, k} [B(s+\lambda(1-B(s)))]^{a_1} \\
&\quad (\beta_1 + \alpha_1 B(s))^{a_1} \prod_{i=1}^k \left(\frac{\beta_{i+1}}{B(s)} + \alpha_{i+1} \right)^{n_i - r_{1,i}} \\
&= e^{-s\tau_1} B(s) \cdot B(s+\lambda(1-B(s))) \cdot (\beta_1 B(s+\lambda(1-B(s)))) + \\
&\quad + \alpha_1 B(s) B(s+\lambda(1-B(s))) \prod_{i=1}^k (\beta_{i+1} \cdot B(s+\lambda(1-B(s)))) + \\
&\quad + \alpha_{i+1} B(s) \cdot B(s+\lambda(1-B(s+\lambda(1-B(s))))^{n_i - r_{1,i}}
\end{aligned}$$

Introducing, as before,

$$B_1(s) = B(s)$$

$$B_2(s) = B(s+\lambda(1-B(s)))$$

we get

$$\begin{aligned}
&e^{-s\tau_1} B_1(s) B_2(s) \cdot (\beta_1 B_2(s) + \alpha_1 B_1(s) B_2(s))^{a_1} \cdot \\
&\cdot \prod_{i=1}^k (\beta_{i+1} B_2(s) + \alpha_{i+1} B_1(s) B_2(s))^{n_i - r_{1,i}}
\end{aligned}$$

Removal of the condition on a_1 yields

$$\begin{aligned}
&E\{e^{-sT_3} | \tilde{n}, \tilde{r}_{1,k}, \tau_1\} = \\
&= e^{-\tau_1(s+\lambda(1-\beta_1 B_2(s)-\alpha_1 B_1(s) B_2(s)))} \cdot B_1(s) \cdot B_2(s) \cdot \\
&\cdot \prod_{i=1}^k (\beta_{i+1} B_2(s) + \alpha_{i+1} B_1(s) B_2(s))^{n_i - r_{1,i}}
\end{aligned}$$

The condition on $r_{1,i}$ can be removed in the same way as that on $r_{2,1}$, and the corresponding sum will be

$$\begin{aligned} & \sum_{r_{1,i}=0}^{n_i} \binom{n_i}{r_{1,i}} \beta_i^{r_{1,i}} \alpha_i^{n_i-r_{1,i}} (\beta_{i+1} B_2(s) + \alpha_{i+1} B_1(s) B_2(s))^{-r_{1,i}} \\ &= \sum_{r_{1,i}=0}^{n_i} \left(\frac{\beta_i}{\beta_{i+1} B_2(s) + \alpha_{i+1} B_1(s) B_2(s)} \right)^{r_{1,i}} \cdot \alpha_i^{n_i-r_{1,i}} \\ &= \left(\frac{\beta_i}{\beta_{i+1} B_2(s) + \alpha_{i+1} B_1(s) B_2(s)} + \alpha_i \right)^{n_i} \end{aligned}$$

and consequently

$$\begin{aligned} E\{e^{-sT_3} | \tilde{n}, \tau_1\} &= e^{-\tau_1 (s + \lambda (1 - \beta_1 B_2(s) - \alpha_1 B_1(s) B_2(s)))} \cdot B_1(s) B_2(s) \\ &\quad \cdot \prod_{i=1}^k (\beta_i + \alpha_i \beta_{i+1} B_2(s) + \alpha_i \alpha_{i+1} B_1(s) B_2(s))^{n_i} \end{aligned}$$

$$\text{with } E\{e^{-s\tau_1}\} = [B(s)]^{n+1}$$

we find that

$$E\{e^{-sT_3} | \tilde{n}\} = (B(s + \lambda (1 - \beta_1 B_2(s) - \alpha_1 B_1(s) B_2(s))))^{n+1} \cdot$$

$$\cdot B_1(s) \cdot B_2(s) \cdot \prod_{i=1}^k (\beta_i + \alpha_i \beta_{i+1} B_2(s) + \alpha_i \alpha_{i+1} B_1(s) B_2(s))^{n_i}$$

writing

$$B_3(s) = B(s + \lambda(1 - \beta_1 B_2(s) - \alpha_1 B_1(s) \cdot B_2(s)))$$

we have that

$$E\{e^{-sT_3} | \tilde{n}\} = B_1(s) \cdot B_2(s) \cdot B_3(s) \cdot$$

$$\prod_{i=1}^k (\beta_i B_3(s) + \alpha_i \beta_{i+1} B_2(s) B_3(s) + \alpha_i \alpha_{i+1} B_1(s) B_2(s) \cdot B_3(s))^{n_i}$$

and finally, after removing the condition on $\tilde{n}$,

$$E\{e^{-sT_3}\} = B_1(s) \cdot B_2(s) \cdot B_3(s) \cdot Q(t_1, t_2, \dots, t_k)$$

where

$$t_i = \beta_i B_3(s) + \alpha_i \beta_{i+1} B_2(s) B_3(s) + \alpha_i \alpha_{i+1} B_1(s) B_2(s) B_3(s)$$

At this point it is tempting to guess the formula for an arbitrary number of turns.

In order to make the generality apparent, however, let us look at T_4 as well

$$\begin{aligned} & E\{e^{-sT_4} | \tilde{n}, \tilde{x}_{1,k}, \tau_1, a_1, \tilde{x}_{2,k-1}, \tau_2, a_2, \tilde{x}_{3,k-2}, \tau_3, a_3, s_{11}, s_{12}, s_{21}\} \\ &= e^{-s(\tau_1 + \tau_2 + \tau_3)} [B(s)]^{n+1-R_{1,k}-R_{2,k-1}-R_{3,k-2}+a_1+a_2+a_3-s_{11}-s_{12}-s_{21}} \end{aligned}$$

Remove the condition on a_3

$$\begin{aligned} & E\{e^{-sT_4} | \tilde{n}, \tilde{x}_{1,k}, \tau_1, a_1, \tilde{x}_{2,k-1}, \tau_2, a_2, \tilde{x}_{3,k-2}, \tau_3, s_{11}, s_{12}, s_{21}\} = \\ &= e^{-s(\tau_1 + \tau_2 + \tau_3)} [B(s)]^{n+1-R_{1,k}-R_{2,k-1}-R_{3,k-2}+a_1+a_2-s_{11}-s_{12}-s_{21}} \end{aligned}$$

$$\cdot e^{-\lambda \tau_3 (1-B(s))}$$

s_{21} can range from zero to a_2 , and the following sum will appear in removal of that condition

$$\begin{aligned} & \sum_{s_{21}=0}^{a_2} \binom{a_2}{s_{21}} \beta_1^{s_{21}} \alpha_1^{a_2-s_{21}} \cdot [B(s)]^{-s_{21}} \\ &= \left(\frac{\beta_1}{B(s)} + \alpha_1 \right)^{a_2} \end{aligned}$$

and we get

$$\begin{aligned} & E\{e^{-sT_4} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}, \tau_2, a_2, \tilde{r}_{3,k-2}, \tau_3, s_{11}, s_{12}\} = \\ &= e^{-s(\tau_1 + \tau_2 + \tau_3)} [B(s)]^{n+1-R_1, k-R_2, k-1-R_3, k-2+a_1-s_{11}-s_{12}} \\ & \cdot e^{-\lambda \tau_3 (1-B(s))} (\beta_1 + \alpha_1 B(s))^{a_2} \end{aligned}$$

s_{12} can be treated in a similar fashion, but we have to remember that it is the second time the remaining a_1-s_{11} customers have option to leave. This probability is $1-\alpha_2 = \beta_2$.

Thus we calculate

$$\sum_{s_{12}}^{a_1-s_{11}} \binom{a_1-s_{11}}{s_{12}} \beta_2^{s_{12}} \alpha_2^{a_1-s_{11}-s_{12}} \cdot [B(s)]^{-s_{12}} = \left(\frac{\beta_2}{B(s)} + \alpha_2 \right)^{a_1-s_{11}}$$

and consequently

$$\begin{aligned} & E\{e^{-sT_4} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}, \tau_2, a_2, \tilde{r}_{3,k-2}, \tau_3, s_{11}\} \\ &= e^{-s(\tau_1 + \tau_2 + \tau_3)} [B(s)]^{n+1-R_1, k-R_2, k-1-R_3, k-2} \end{aligned}$$

$$\cdot e^{-\lambda \tau_3 (1-B(s))} (\beta_1 + \alpha_1 B(s))^{a_2} (\beta_2 + \alpha_2 B(s))^{a_1 - s_{11}}$$

The Laplace transform of the density function of τ_3 is

$$B(s)^{n+1-R_1, k-R_2, k-1+a_1+a_2-s_{11}}$$

and removal of the condition on τ_3 yields

$$\begin{aligned} & E\{e^{-sT_4} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}, \tau_2, a_2, \tilde{r}_{3,k-2}, s_{11}\} = \\ & = e^{-s(\tau_1 + \tau_2)} [B(s)]^{n+1-R_1, k-R_2, k-1-R_3, k-2} \\ & \quad \cdot [B(s + \lambda(1-B(s)))]^{n+1-R_1, k-R_2, k-1+a_1+a_2-s_{11}} \\ & \quad \cdot (\beta_1 + \alpha_1 B(s))^{a_2} \cdot (\beta_2 + \alpha_2 B(s))^{a_1 - s_{11}} \\ & = e^{-s(\tau_1 + \tau_2)} \cdot B_1(s) \cdot B_2(s) [B_1(s) B_2(s)]^{n-R_1, k-R_2, k-1} \\ & \quad \cdot (\beta_1 B_2(s) + \alpha_1 B_1(s) B_2(s))^{a_2} \cdot (\beta_2 B_2(s) + \alpha_2 B_1(s) B_2(s))^{a_1 - s_{11}} \\ & \quad \cdot [B_1(s)]^{-R_3, k-2} \end{aligned}$$

with $B_1(s)$ and $B_2(s)$ as before.

The condition on s_{11} can be removed by using the following sum

$$\sum_{s_{11}=0}^{a_1} \binom{a_1}{s_{11}} \beta_1^{s_{11}} \alpha_1^{a_1 - s_{11}} (\beta_2 B_2(s) + \alpha_2 B_1(s) B_2(s))^{-s_{11}} =$$

$$\begin{aligned}
&= \sum_{s_{11}=0}^{a_1} \binom{a_1}{s_{11}} \left(\frac{\beta_1}{\beta_2 B_2(s) + \alpha_2 B_1(s) B_2(s)} \right)^{s_{11}} \alpha_1^{a_1 - s_{11}} \\
&= \left(\frac{\beta_1}{\beta_2 B_2(s) + \alpha_2 B_1(s) B_2(s)} + \alpha_1 \right)^{a_1}
\end{aligned}$$

and from this

$$\begin{aligned}
&E\{e^{-sT_4} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, r_{2,k-1}, \tau_2, a_2, r_{3,k-2}\} \\
&= e^{-s(\tau_1 + \tau_2)} B_1(s) \cdot B_2(s) [B_1(s) \cdot B_2(s)]^{n-R_1, k-R_2, k-1} \\
&\quad \cdot (\beta_1 B_2(s) + \alpha_1 B_1(s) B_2(s))^{a_2} [B_1(s)]^{-R_3, k-2} \\
&\quad \cdot (\beta_1 + \alpha_1 \beta_2 B_2(s) + \alpha_1 \alpha_2 B_1(s) \cdot B_2(s))^{a_1}
\end{aligned}$$

The elements of $\tilde{r}_{3,k-2}$ can range from zero to $\tilde{n} - \tilde{r}_{1,k} - \tilde{r}_{2,k-1}$ and we arrive at the following sum

$$\begin{aligned}
&\sum_{r_{3,i}=0}^{n_i - r_{1,i} - r_{2,i}} \binom{n_i - r_{1,i} - r_{2,i}}{r_{3,i}} \beta_{i+2}^{r_{3,i}} \cdot \alpha_{i+2}^{n_i - r_{1,i} - r_{2,i} - r_{3,i}} \\
&\quad \cdot [B_1(s)]^{-r_{3,i}} \\
&= \left(\frac{\beta_{i+2}}{B_1(s)} + \alpha_{i+2} \right)^{n_i - r_{1,i} - r_{2,i}}
\end{aligned}$$

which yields

$$E\{e^{-sT_4} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}, \tau_2, a_2\} =$$

$$\begin{aligned}
&= e^{-s(\tau_1 + \tau_2)} B_1(s) \cdot B_2(s) \prod_{i=1}^k (\beta_{i+2} B_2(s) + \\
&\quad + \alpha_{i+2} B_1(s) \cdot B_2(s))^{n_i - r_{1,i} - r_{2,i}} (\beta_1 B_2(s) + \alpha_1 B_1(s) B_2(s))^{a_2} \\
&\quad \cdot (\beta_1 + \alpha_1 \beta_2 B_2(s) + \alpha_1 \alpha_2 B_1(s) B_2(s))^{a_1}
\end{aligned}$$

The condition on a_2 can now be removed and we get

$$\begin{aligned}
&E\{e^{-sT_4} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}, \tau_2\} = \\
&= e^{-s(\tau_1 + \tau_2)} B_1(s) B_2(s) \\
&\quad \cdot \prod_{i=1}^k (\beta_{i+2} B_2(s) + \alpha_{i+2} B_1(s) B_2(s))^{n_i - r_{1,i} - r_{2,i}} \\
&\quad \cdot e^{-\lambda(1 - \beta_1 B_2(s) + \alpha_1 B_1(s) B_2(s))} \\
&\quad \cdot (\beta_1 + \alpha_1 \beta_2 B_2(s) + \alpha_1 \alpha_2 B_1(s) B_2(s))^{a_1}
\end{aligned}$$

Now,

$$E\{e^{-s\tau_2}\} = [B(s)]^{n+1-R_1, k+a_1}$$

and consequently

$$\begin{aligned}
&E\{e^{-sT_4} | \tilde{n}, \tilde{r}_{1,k}, \tau_1, a_1, \tilde{r}_{2,k-1}\} = \\
&= e^{-s\tau_1} B_1(s) \cdot B_2(s) \prod_{i=1}^k (\beta_{i+2} B_2(s) + \alpha_{i+2} B_1(s) \cdot B_2(s))^{n_i - r_{1,i} - r_{2,i}}
\end{aligned}$$

$$\begin{aligned}
& \cdot (B(s + \lambda(1 - \beta_1 B_2(s) + \alpha_1 B_1(s) B_2(s))))^{n+1-R_1, k+a_1} \\
& \cdot (\beta_1 + \alpha_1 \beta_2 B_2(s) + \alpha_1 \alpha_2 B_1(s) B_2(s))^{a_1} \\
& = e^{-s\tau_1} B_1(s) \cdot B_2(s) \cdot B_3(s) \\
& \cdot (\beta_1 B_3(s) + \alpha_1 \beta_2 B_2(s) + \alpha_1 \alpha_2 B_1(s) B_2(s) B_3(s))^{a_1} \\
& \cdot \prod_{i=1}^k (\beta_{i+2} B_2(s) B_3(s) + \alpha_{i+2} B_1(s) B_2(s) B_3(s))^{n_i - r_{1,i}} \\
& \cdot (\beta_{i+2} B_2(s) + \alpha_{i+2} B_1(s) B_2(s))^{-r_{2,i}}
\end{aligned}$$

As before, the condition on $r_{2,i}$ is removed, and the following sum has to be calculated

$$\begin{aligned}
& \sum_{r_{2,i}}^{n_i - r_{1,i}} \binom{n_i - r_{1,i}}{r_{2,i}} \beta_{i+1}^{r_{2,i}} \alpha_{i+1}^{n_i - r_{1,i} - r_{2,i}} \cdot \\
& \cdot (\beta_{i+2} B_2(s) + \alpha_{i+2} B_1(s) B_2(s))^{-r_{2,i}} \\
& \left(\frac{\beta_{i+1}}{\beta_{i+2} B_2(s) + \alpha_{i+2} B_1(s) B_2(s)} + \alpha_{i+1} \right)^{n_i - r_{1,i}}
\end{aligned}$$

and

$$\begin{aligned}
& E\{e^{-sT_4} | \tilde{n}, \tilde{x}_1, k, \tau_1, a_1\} = \\
& = e^{-s\tau_1} \cdot B_1(s) \cdot B_2(s) \cdot B_3(s) \cdot
\end{aligned}$$

$$\begin{aligned}
& \cdot \prod_{i=1}^k (\beta_{i+1} B_3(s) + \alpha_{i+1} \beta_{i+2} B_2(s) \cdot B_3(s) + \\
& + \alpha_{i+1} \alpha_{i+2} B_1(s) B_2(s) B_3(s))^{n_i - r_{1,i}} \\
& \cdot (\beta_1 B_3(s) + \alpha_1 \beta_2 B_2(s) B_3(s) + \alpha_1 \alpha_2 B_1(s) \cdot B_2(s) \cdot B_3(s))^{a_1}
\end{aligned}$$

Removing the condition on a_1 yields

$$\begin{aligned}
& E\{e^{-sT_4} | \tilde{n}, \tilde{r}_{1,k}, \tau_1\} = \\
& = e^{-s\tau_1} B_1(s) \cdot B_2(s) \cdot B_3(s) \prod_{i=1}^k (\beta_{i+1} B_3(s) + \\
& + \alpha_{i+1} \beta_{i+2} B_2(s) \cdot B_3(s) + \alpha_{i+1} \alpha_{i+2} B_1(s) \cdot B_2(s) \cdot B_3(s))^{n_i - r_{1,i}} \\
& \cdot e^{-\lambda(1 - \beta_1 B_3(s) - \alpha_1 \beta_2 B_2(s) B_3(s) - \alpha_1 \alpha_2 B_1(s) \cdot B_2(s) \cdot B_3(s))}
\end{aligned}$$

$$\text{but } E\{e^{-s\tau_1}\} = [B(s)]^{n+1}$$

and thus

$$\begin{aligned}
& E\{e^{-sT_4} | \tilde{n}, \tilde{r}_{1,k}\} = \\
& = B_1(s) \cdot B_2(s) \cdot B_3(s) \cdot B_4(s) \prod_{i=1}^k (\beta_{i+1} B_3(s) + \\
& + \alpha_{i+1} \beta_{i+2} B_2(s) B_3(s) + \alpha_{i+1} \alpha_{i+2} B_1(s) \cdot B_2(s) \cdot B_3(s))^{n_i - r_{1,i}} \\
& \cdot [B_4(s)]^{n_i}
\end{aligned}$$

where

$$B_4(s) = B(s + \lambda(1 - \beta_1 B_3(s) - \alpha_1 \beta_2 B_2(s) B_3(s) - \alpha_1 \alpha_2 B_1(s) B_2(s) B_3(s)))$$

The sum

$$\sum_{r_{1,i}=0}^{n_i} \binom{n_i}{r_{1,i}} \beta_i^{r_{1,i}} \alpha_i^{n_i - r_{1,i}}.$$

$$\cdot (\beta_{i+1} B_3(s) + \alpha_{i+1} \beta_{i+2} B_2(s) B_3(s) +$$

$$+ \alpha_{i+1} \alpha_{i+2} B_1(s) B_2(s) B_3(s))^{-r_{1,i}}$$

$$= \left(\frac{\beta_i}{\beta_{i+1} B_3(s) + \alpha_{i+1} \beta_{i+2} B_2(s) B_3(s) + \alpha_{i+1} \alpha_{i+2} B_1(s) B_2(s) B_3(s)} + \alpha_i \right)^{n_i}$$

and we get

$$E\{e^{-sT_4} | \tilde{n}\} = B_1(s) B_2(s) B_3(s) B_4(s) \prod_{i=1}^k (\beta_i B_4(s) +$$

$$+ \alpha_i \beta_{i+1} B_3(s) B_4(s) + \alpha_i \alpha_{i+1} \beta_{i+2} B_2(s) B_3(s) B_4(s) +$$

$$+ \alpha_i \alpha_{i+1} \alpha_{i+2} B_1(s) B_2(s) B_3(s) B_4(s))^{n_i}$$

and after removal of the condition on $\tilde{n}$

$$E\{e^{-sT_4}\} =$$

$$B_1(s) B_2(s) B_3(s) B_4(s) \cdot Q(t_1, t_2, \dots, t_k)$$

where

$$t_i = \beta_i B_4(s) + \alpha_i \beta_{i+1} B_3(s) B_4(s) + \alpha_i \alpha_{i+1} \beta_{i+2} B_2(s) B_3(s) \cdot$$

$$\cdot B_4(s) + \alpha_i \alpha_{i+1} \alpha_{i+2} B_1(s) B_2(s) B_3(s) B_4(s)$$

It would be possible to continue this process indefinitely, but it is obvious that the general solution can be written down at this stage. If we list the four first transforms, we find that

$$E\{e^{-sT_1}\} = B_1(s) Q(t_1^{(1)}, t_2^{(1)}, \dots, t_k^{(1)})$$

$$E\{e^{-sT_2}\} = B_1(s) B_2(s) Q(t_1^{(2)}, t_2^{(2)}, \dots, t_k^{(2)})$$

$$E\{e^{-sT_3}\} = B_1(s) B_2(s) B_3(s) Q(t_1^{(3)}, t_2^{(3)}, \dots, t_k^{(3)})$$

$$E\{e^{-sT_4}\} = B_1(s) B_2(s) B_3(s) B_4(s) Q(t_1^{(4)}, t_2^{(4)}, \dots, t_k^{(4)})$$

where

$$B_1(s) = B(s)$$

$$B_2(s) = B(s + \lambda(1 - B_1(s)))$$

$$B_3(s) = B(s + \lambda(1 - \beta_1 B_2(s) - \alpha_1 B_1(s) B_2(s)))$$

$$B_4(s) = B(s + \lambda(1 - \beta_1 B_3(s) - \alpha_1 \beta_2 B_1(s) B_2(s) - \alpha_1 \alpha_2 B_1(s) B_2(s) B_3(s)))$$

$$t_i^{(1)} = B_1(s)$$

$$t_i^{(2)} = \beta_i B_2(s) + \alpha_i B_1(s) B_2(s)$$

$$t_i^{(3)} = \beta_i B_3(s) + \alpha_i \beta_{i+1} B_2(s) B_3(s) + \alpha_i \alpha_{i+1} B_1(s) B_2(s) B_3(s)$$

$$t_i^{(4)} = \beta_i B_4(s) + \alpha_i \beta_{i+1} B_3(s) B_4(s) + \alpha_i \alpha_{i+1} \beta_{i+2} \cdot$$

$$\cdot B_2(s) B_3(s) B_4(s) + \alpha_i \alpha_{i+1} \alpha_{i+2} B_1(s) B_2(s) B_3(s) B_4(s)$$

It is apparent from the structure of the calculations that the transform of the response time up to and including the

n :th service of the "tagged" customer can be written as

$$E\{e^{-sT_n}\} = B_1(s) \cdot B_2(s) \dots B_n(s) \cdot Q(t_1^{(n)}, t_2^{(n)}, \dots, t_k^{(n)})$$

where

$$B_1(s) = B(s) = \frac{\mu}{s + \mu}$$

In order to compress the notation, we introduce the following auxiliary functions

$$\alpha_i(j) = \begin{cases} 1 & j = 0 \\ \prod_{m=0}^{j-1} \alpha_{i+m} & j > 0 \end{cases}$$

$$\beta_m(i, j) = \begin{cases} 1 & j = m \\ \beta_{i+j} & j < m \end{cases}$$

and

$$S(p, i, s) = \sum_{j=0}^{p-1} \alpha_i(j) \beta_{p-1}(i, j) \prod_{r=0}^j B_{p-j}(s)$$

This means that $B_n(s)$ can be written as

$$B_n(s) = B(s + \lambda(1 - S(n-1, 1, s)))$$

and that

$$t_i^{(n)} = S(n, i, s)$$

The formula above is based on conjectures. A proper proof has so far not been possible to find, but there are a number of circumstances that talk in favour of the formula. An alternative recursive formula has been derived by D Rapp and it can be shown that the two formulae agree in some special cases that have been possible to compare. The mean response time has been calculated and shown to be in agreement with results obtained from queueing network theory and Little's result.

